Multivariate Statistical Process Evaluation and Monitoring for Complex Chemical Processes

by

Ruan Francois Rossouw

Dissertation presented for the degree of

Doctor of Philosophy in Statistics

at the Stellenbosch University

Department of Statistics and Actuarial Science,
University of Stellenbosch,
Private Bag X1, 7602 Matieland, South Africa

Promoters:

Prof. N.J. Le Roux
Department of Statistics and Actuarial Science
University of Stellenbosch

Dr. R.L.J. Coetzer
Industrial Statistics
Group Technology R&T
Sasol

December 2015

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Ruan Rossouw

Date: 29-October-2015

Abstract

Multivariate Statistical Process Evaluation and Monitoring for Complex Chemical Processes

R.F. Rossouw

Department of Statistics and Actuarial Science,
University of Stellenbosch,
Private Bag X1, 7602 Matieland, South Africa

Dissertation: PhD Statistics

December 2015

In this study, the development of an innovative fully integrated process monitoring methodology is presented for a complex chemical facility, originating at the coal feed from different mines up to the processing of the coal to produce raw gas at the gasification plant. The methodology developed is real-time, visual, detect deviations from expected performance across the whole value chain, and also provide for the integration and standardisation of data from a number of different data sources and formats.

Real time coal quality analyses from an XRF analyser are summarised and integrated with various data sources from the Coal Supply Facility to provide information on the coal quality of each mine. In addition, simulation models are developed to generate information on the coal quality of each heap and the quality of the reclaimed coal sent to gasification.

A real-time multivariate process monitoring approach for the Coal Gasification Facility is presented. This includes a novel approach utilising Generalised Orthogonal Procrustes Analysis to find the optimal units and time period to employ as a reference set. Principal Component Analysis (PCA) and Canonical Variate Analysis (CVA) theory and biplots are evaluated and extended for the real-time monitoring of the plant.

A new approach to process deviation monitoring on many variables is presented based on the confidence (α) value at a specified T^2-value. This methodology is proposed as a general data driven performance index as it is objective, and very little prior knowledge of the system is required.

A new multivariate gasifier performance index (GPI) is developed, which integrates subject matter knowledge with a data driven approach for real time

performance monitoring. Various software modules are developed which were required for the implementation of the real time multivariate process monitoring methodology, which is made operational and distributed to the clients on an interactive web interface. The methodology has been trademarked by Sasol as the MSPEM™ Technology Package. Following the success of the developed methodology, the MSPEM™ package has been rolled out to many more business units within the Sasol Group.

In conclusion, this study presents the development and implementation of the MSPEM™ application for a real-time, integrated and standardised approach to multivariate process monitoring of the Sasol Synfuels Coal Value Chain and Gasification Facility. In summary, the following novel developments were introduced:

- The application of distance measures other than Euclidean measures are introduced for space filling designs for computer experiments in mixture variables.
- An approach utilising Generalised Orthogonal Procrustes Analysis to specify the optimal units and time period to employ as a reference set is developed.
- An approach to process deviation monitoring on many variables is presented based on the confidence (α) value at a specified T^2-value.
- An integrated approach to a reactor performance index is developed and illustrated.
- A comprehensive software infrastructure is developed and implemented.

Uittreksel

Multivariate Statistical Process Evaluation and Monitoring for Complex Chemical Processes

R.F. Rossouw

Department of Statistics and Actuarial Science,
University of Stellenbosch,
Private Bag X1, 7602 Matieland, South Africa

Proefskrif: PhD Statistics

Desember 2015

In hierdie studie word die ontwikkeling van 'n innoverende en ten volle geïntegreerde proses moniteringsmetodologie vir 'n komplekse chemiese fasiliteit aangebied. Die metodologie is ontwikkel vir die monitering van die steenkool kwaliteite vanaf die verskillende myne tot en met die verwerking van die steenkool om rou gas te produseer by die steenkool vergassingsaanleg, asook die intydse monitering van die gasproduksie en effektiwiteit van die aanleg. Die ontwikkelde metode is intyds, visueel, spoor afwykings van verwagte verrigting oor die hele waarde ketting op, en maak ook voorsiening vir die integrasie en standaardisering van data afkomstig van verskillende bronne en formate.

Intydse steenkool kwaliteitsmetings met 'n XRF analiseerder word opgesom en geïntegreer met verskeie bestaande data bronne uit die steenkoolfasiliteit om inligting oor die gehalte van steenkool vanaf elke myn te voorsien. Daarbenewens is simulasie modelle ontwikkel om inligting oor die kwaliteit van elke steenkool bergingshoop sowel as die kwaliteit van die herwonne steenkool na vergassing te verskaf.

'n Intydse meerveranderlike proses moniteringsmetodologie vir die steenkool vergassingsfasiliteit word aangebied. Dit sluit in 'n nuwe benadering om die optimale reaktors en tydperk te vind wat gebruik kan word as die verwysingsdatastel. Veralgemeende Ortogonale Procrustes Analise (GOPA) is hiervoor gebruik en aangepas. Hoofkomponent-analise (PCA) en Kanoniese Veranderlike Analise (CVA) teorie, tesame met bistippings, word geëvalueer en uitgebrei vir die intydse monitering van die produksieaanleg.

'n Nuwe benadering tot die monitering van die gelyktydige proses afwykings van 'n groot aantal veranderlikes word aangebied, gebaseer op die vertrouenskoëffisiënt (α) vir 'n bepaalde T^2-waarde. Hierdie metodologie word voorgestel

as 'n algemene data-gedrewe verrigtingsindeks aangesien dit objektief is, en baie min historiese kennis van die stelsel word vereis.

'n Nuwe meerveranderlike verrigtingsindeks (GPI) vir die vergassers is ontwikkel, wat kennis van die proses integreer met 'n data-gedrewe benadering vir die intydse monitering van verrigting. Verskeie sagteware modules is ontwikkel vir die implementering van die intydse meerveranderlike prosesmoniteringsmetodologie, wat operasioneel gemaak en beskikbaar gestel is aan die kliënte met behulp van 'n interaktiewe webkoppelvlak. Die metodologie is gehandelsmerk deur Sasol as die MSPEM™ Tegnologie Pakket. Na aanleiding van die sukses van die ontwikkelde metodologie, is die MSPEM™ pakket uitgerol na baie meer produksie aanlegte in Sasol.

Ten slotte, hierdie studie bied die ontwikkeling en implementering van die MSPEM™ pakket aan vir 'n intydse, geïntegreerde en gestandaardiseerde benadering tot meerveranderlike proses monitering van die Sasol Synfuels Steenkool Waardeketting en die Steenkool Vergassingsfasiliteit. Verder is die volgende nuwe ontwikkelings bekendgestel:

- Die toepassing van afstandsmetings anders as Euklidiese afstand om die eksperimentele ruimte te vul in rekenaareksperimente.
- 'n Benadering is ontwikkel om die optimale reaktors en tydperk te vind wat gebruik kan word as 'n verwysingsdatastel vir intydse monitering, deur gebruik te maak van Veralgemeende Ortogonale Procrustes Analise (GOPA).
- 'n Benadering gebaseer op die vertrouenskoëffisiënt (α) vir 'n bepaalde T^2-waarde is ontwikkel vir die monitering van die gelyktydige proses afwykings van 'n groot aantal veranderlikes.
- 'n Geïntegreerde benadering is ontwikkel vir die verkryging van 'n reaktor verrigtingsindeks en is kommersieël toegepas en geïllustreer.

Contents

Declaration i

Abstract ii

Uittreksel iv

Contents vi

List of Figures viii

List of Tables xiv

1 Introduction 1
- 1.1 A Complex Chemical Plant . . . 1
- 1.2 Problem Setting . . . 4
- 1.3 Research Objectives Breakdown . . . 9
- 1.4 Thesis Outline . . . 14
- 1.5 Acronyms . . . 15

2 Sasol Coal Supply 16
- 2.1 Data Sources . . . 17
- 2.2 X-Ray Fluorescence analyzer (XRF) . . . 25
- 2.3 Stacker Simulation Model . . . 26
- 2.4 Reclaimer Simulation . . . 44
- 2.5 Empirical Slag Model . . . 51
- 2.6 Conclusions . . . 64

3 Coal Gasification 66
- 3.1 Coal Gasification Monitoring . . . 67
- 3.2 Reference set Selection . . . 68
- 3.3 Multivariate process monitoring using the PCA biplot . . . 90
- 3.4 CVA Biplots for Monitoring . . . 117
- 3.5 Discussion . . . 133

4 Gasifier Performance Index 136

4.1 Fundamental Gasifier Performance Index (GPI) . . . 136
4.2 Empirical Gasifier Performance Index . . . 145
4.3 Integration of fundamental and empirical approaches . . . 161
4.4 Comparison of GPI indices . . . 193
4.5 Conclusions . . . 200

5 Software Infrastructure 204
5.1 Distributed Control System (DCS) Data Interface . . . 206
5.2 Date/Time utility functions . . . 210
5.3 Data Storage . . . 211
5.4 Generic Database Interface Functions . . . 212
5.5 Local DCS data interface functions . . . 214
5.6 Statistical Programming . . . 226
5.7 User Interface . . . 245
5.8 Conclusions . . . 252

6 Conclusions and Future Research 255
6.1 Conclusions . . . 255
6.2 Future Research . . . 265

Appendices 267

A Data Interface Software 269
A.1 PI API interface functions (`sslpiutils` and `sslgpipi`) . . . 269
A.2 Date Conversion Functions (`ssldtutils`) . . . 288
A.3 Database Interface Functions (`ssldbutils`) . . . 294
A.4 Local DCS Data Interface Functions (`ssldcsutils`) . . . 309
A.5 Excel Interface Functions (`sslxlutils`) . . . 340

B Statistical Software 346
B.1 Code Listings for the `mltv` package . . . 346
B.2 Code Listings for GOPA . . . 380

C User Interface 386
C.1 Sasol Coal Supply Interface . . . 386
C.2 Sasol Gasification and Coal Value Chain Interface . . . 396

List of References 402

List of Figures

1.1 Secunda Coal Value Chain (CVC) overview . . . 2
1.2 Gasification facility overview . . . 3
1.3 Research objectives overview . . . 5
1.4 Software infrastructure overview . . . 9

2.1 Secunda Coal Supply (SCS) overview . . . 18
2.2 Secunda Coal Supply (SCS) data overview . . . 19
2.3 XRF Spectra table . . . 21
2.4 XRF Results table . . . 22
2.5 Sasol Coal Supply (SCS) Material Movement file . . . 22
2.6 Sasol Coal Supply (SCS) Material Movement file for a specific heap ID . . . 23
2.7 Sasol Coal Supply (SCS) Material Movement file with converted date and time . . . 23
2.8 Sasol Coal Supply (SCS) Coal Stacking and Reclaiming Picture (CSRP) . . . 24
2.9 Sasol Coal Supply (SCS) stockpile information data . . . 25
2.10 X-Ray Fluorescence Analyzer (XRF) . . . 26
2.11 Position of XRF on conveyor to Stacker 4 . . . 27
2.12 Stack yard . . . 28
2.13 Stacking and reclaiming . . . 28
2.14 Stacker . . . 29
2.15 Reclaimer . . . 30
2.16 Histogram of mine percentages to Stacker 4 . . . 32
2.17 Histogram of mine percentages to Stacker 1 . . . 33
2.18 Table of mapped ash and mine data . . . 35
2.19 Histogram of mine percentages on Heap 15374 . . . 37
2.20 Ash Percentage over length for Heap 15374 . . . 41
2.21 Overall as well as ROM ash distributions . . . 42
2.21 Overall as well as ROM ash distributions continued . . . 43
2.22 Mine layers over length for Heap 15374 . . . 45
2.23 Predicted reclaimed coal ash percentage . . . 52
2.24 Algorithmic Overview of Optimization Methodology . . . 56

2.25 Ternary plot of the optimal Minimax design using Aitchison distance as dissimilarity criterion 60
2.26 Ternary plot of the optimal Minimax design using Divergence as dissimilarity criterion 60
2.27 Ternary plot of the optimal Minimax design using Euclidean distance as dissimilarity criterion 61
2.28 Scaled Component Values plot of maximin design for ash composition 62
2.29 Actual versus predicted plot for slag prediction model 63
2.30 Ternary plot the predicted slag percent as a function of the first three ash composition variables 63

3.1 Gasification Overview 66
3.2 CVA Biplots for Eastern Factory for a one week period 73
3.2 CVA Biplots for Eastern Factory for a one week period continued . 74
3.3 GOPA data structure 77
3.4 Average monthly GOPA sum of squares contribution for Western factory (the optimal two months are highlighted in red) 79
3.5 Average monthly GOPA sum of squares contribution for Eastern factory (the optimal two months are highlighted in red) 80
3.6 Isotropic scaling factors for optimal one and two month combinations for the Eastern factory 81
3.7 PCA plots of group averages including the variation for each train for the optimal one and two month combinations (the overall centroid $\boldsymbol{O}$ is depicted at the intersection of the two lines on each plot) 83
3.7 PCA plots of group averages including the variation for each train for the optimal one and two month combinations (the overall centroid $\boldsymbol{O}$ is depicted at the intersection of the two lines on each plot) 84
3.8 Data structure for calculating the group average matrix, $\boldsymbol{G}$. Variables named according to Table 3.1 85
3.9 PCA biplot of group averages with variable axes for the optimal month for the Eastern factory 86
3.10 PCA biplot of group averages with variable axes for the optimal month for the Western factory 87
3.11 Steps for determining the optimal reference set for multivariate monitoring of multiple parallel processes 89
3.12 Scree Plot 96
3.13 Cumulative percentage variance explained 97
3.14 Permutation versus original eigenvalues (The permuted values are depicted in the box plot, and the original values are highlighted in red) 97
3.15 Permutation versus original variance explained (The permuted values are depicted in the box plot, and the original values are highlighted in red) 98

3.16 PCA plots of different principal component combinations for the reference set . . . 103
3.16 PCA plots of different principal component combinations for the reference set continued . . . 104
3.17 PCA plots of different principal component combinations with axes chosen according to the axis predictivity criterion . . . 105
3.17 PCA plots of different principal component combinations with axes chosen according to the axis predictivity criterion continued . . . 106
3.18 PCA plots of different principal component combinations with axes chosen according to the MSPE criterion . . . 107
3.18 PCA plots of different principal component combinations with axes chosen according to the MSPE criterion continued . . . 108
3.19 PCA plots of different principal component combinations and monitoring ellipses . . . 111
3.19 PCA plots of different principal component combinations and monitoring ellipses continued . . . 112
3.19 PCA plots of different principal component combinations and monitoring ellipses continued . . . 113
3.20 Steps for the implementation of a PCA biplot monitoring methodology . . . 115
3.21 CVA Biplot of a production train . . . 118
3.22 Scree plot . . . 123
3.23 Cumulative percentage variance explained . . . 124
3.24 CVA biplots for the different combinations of PC's . . . 128
3.24 CVA biplots for the different combinations of PC's continued . . . 129
3.24 CVA biplots for the different combinations of PC's continued . . . 130
3.25 Steps for the implementation of a CVA biplot monitoring methodology . . . 132
3.26 PCA Biplot of first two principal components . . . 134

4.1 Fundamental GPI graph . . . 140
4.2 Zoomed in mini graph for GG26 . . . 140
4.3 Variable contribution graph for GG26 . . . 141
4.4 Trend plots of the process variables for GG26 . . . 143
4.4 Trend plots of the process variables for GG26 continued . . . 144
4.5 Demonstration of the effect of different delta value choices for the minimum and maximum . . . 144
4.6 Scree Plot for reference set ($\boldsymbol{X}$) for Eastern factory . . . 150
4.7 T^2-Statistic for GG26 over time . . . 151
4.8 F-distribution probability values (α confidence on index) for the T^2-values for GG26 . . . 152
4.9 Empirical GPI graph . . . 153
4.10 PCA plots for different principal component combinations for GG26 155

4.10 PCA plots for different principal component combinations for GG26 continued . . . 156
4.11 Variable Contributions . . . 157
4.11 Variable Contributions continued . . . 158
4.12 Proposed methodology for an empirical performance index . . . 159
4.13 Comparison of fundamental and empirical GPI at low factory load conditions . . . 160
4.14 Average monthly GOPA sum of squares contributions for the Western factory . . . 163
4.15 Average monthly GOPA sum of squares contributions for the Eastern factory . . . 164
4.16 PCA plots of group averages including the variation for each train for the optimal one and two month combinations . . . 165
4.17 Scree Plot for reference data ($\boldsymbol{X}_*$) for Eastern factory . . . 167
4.18 PCA plots of different principal component combinations for the reference set, with axes chosen according to the axis predictivity criterion . . . 170
4.18 PCA plots of different principal component combinations for the reference set, with axes chosen according to the axis predictivity criterion continued . . . 171
4.19 Comparison of fundamental and integrated GPI . . . 172
4.20 PCA plots of different principal component combinations for GG26 175
4.20 PCA plots of different principal component combinations for GG26 continued . . . 176
4.21 Trend plot for GG26 variable U2 . . . 177
4.22 Trend plot for GG26 variable U4 . . . 177
4.23 Trend plot for GG26 variable S7 . . . 177
4.24 Monitoring biplot for with PC2 vs PC3 including all axes for GG26 178
4.25 Variable contribution plot for GG26 at $t = 53$ . . . 179
4.26 PCA plots of different principal component combinations for GG17 180
4.26 PCA plots of different principal component combinations for GG17 continued . . . 181
4.27 Trend plots of the process variables for GG17 . . . 182
4.27 Trend plots of the process variables for GG17 continued . . . 183
4.28 Comparison of fundamental and integrated GPI at low factory load values . . . 183
4.29 PCA plots of different principal component combinations for GG46 184
4.29 PCA plots of different principal component combinations for GG46 continued . . . 185
4.30 Trend plots of the process variables for GG46 . . . 186
4.30 Trend plots of the process variables for GG46 continued . . . 187
4.31 Frequency of variables as one of the two highest contributions to the T^2-value . . . 188
4.32 Scree Plot . . . 190

4.33 Cumulative Percentage Variance Explained 191
4.34 CVA biplots for the Combinations up to 4 Dimensions 192
4.35 Trend plots for GG01 variable U2 highlighting the values with large discrepancies between the fundamental and integrated GPI 196
4.36 Trend plots for GG01 variable S1 highlighting the values with large discrepancies between the fundamental and integrated GPI 197
4.37 Trend plots for GG01 variable S2 highlighting the values with large discrepancies between the fundamental and integrated GPI 197
4.38 Trend plots for GG01 variable S3 highlighting the values with large discrepancies between the fundamental and integrated GPI 198
4.39 Trend plots for GG01 variable S4 highlighting the values with large discrepancies between the fundamental and integrated GPI 198
4.40 Trend plots for GG01 variable S6 highlighting the values with large discrepancies between the fundamental and integrated GPI 199

5.1 MSPEM™ Software Infrastructure Overview 205
5.2 MSPEM™ Architecture Overview 205
5.3 Example for time weighted aggregation (red dot indicate bad value, and green dot indicate interpolated value). 215
5.4 GPI example of SVG Graphic 232
5.5 PCA Biplot demonstration 243
5.6 CVA Biplot demonstration 245

6.1 Steps for determining the optimal reference set for multivariate monitoring of multiple parallel processes 257
6.2 Steps for the implementation of a PCA biplot monitoring methodology 258
6.3 Steps for the implementation of a CVA biplot monitoring methodology 259
6.4 Proposed methodology for an empirical performance index 261
6.5 MSPEM™ Software Infrastructure Overview 262
6.6 MSPEM™ Architecture Overview 264

C.1 SCS Menu structure 386
C.2 SCS Home page overview 387
C.3 SCS Home page ash overview 388
C.4 SCS Home page ash per mine 389
C.5 SCS Home page material movement 390
C.6 SCS Heap Information Menu 391
C.7 SCS Heap Information for Heap 15374 392
C.8 SCS Heap Reclaim Simulation Menu 393
C.9 SCS Heap Reclaim Simulation Heap 15374 393
C.10 SCS Reclaimer Simulation Heap select menu 394
C.11 SCS Reclaimer Simulation Heap information input for Heap 15374 . 394

C.12 SCS Reclaimer Simulation Heap 15374 reclaimed portion 395
C.13 SCS Reclaimer Simulation Output 395
C.14 SCS Heap Report Builder inputs 396
C.15 CVC Menu structure 396
C.16 CVC GPI Menu 397
C.17 CVC GPI East 397
C.18 CVC GPI GG17 Contribution 398
C.19 CVC GPI GG17 Time Series Graphs 399
C.20 CVC GPI GG17 Multivariate Graphs 400
C.21 CVC Long Term Monitoring CVA Graphs 401

List of Tables

1.1 List of acronyms used in this thesis 15

2.1 Material Movement File for Heap 15374 36
2.2 CSRP information for Heap 15374 37
2.3 Mine numbers for Heap 15374 38
2.4 Mine properties for Heap 15374 44
2.5 Summary for Heap 15374 44
2.6 Material movement file for reclaiming 47
2.7 Reclaimed tons from PI DCS system 47
2.8 Material Movement combined with the CSRP data 48
2.9 Reshaped Material Movement and CSRP data 49
2.10 Merged PI and Material Movement File for Western factory 50
2.11 Merged PI and Material Movement File for Eastern factory 51
2.12 Table of generic mixture properties and ranges 59
2.13 Comparison of space filling designs and models for different properties for the generic three variable case. 61
2.14 Ash composition ranges 62

3.1 Variable types 71
3.2 Decomposition of the total sum of squares in orthogonal Procrustes analysis 76
3.3 Optimal combination of Months 80
3.4 Euclidean distance of trains from overall centroid $\boldsymbol{O}$. 81
3.5 Biplot quality of the two-dimensional PCA plot for each side and month combinations (Figures 3.7a - 3.7d) 82
3.6 PCA biplot quality for Figures 3.16a to 3.16f 100
3.7 Axis Predictivity 102
3.8 Axis MSPE 102
3.9 PCA loadings for first four principal components 102
3.10 Axis predictivity values with values above 0.35 highlighted 110
3.11 Table with new data 110
3.12 CVA biplot quality in the original variables for Figures 3.24a to 3.24f 123
3.13 Axis predictivity values 124
3.14 Axes MSPE values 124

3.15 Group mean values . . . 125
3.16 Group standard deviation values . . . 125
3.17 CVA classification accuracy . . . 125

4.1 Variable types . . . 137
4.2 Optimal combination of Months . . . 164
4.3 Euclidean distance of trains from overall centroid $\boldsymbol{O}$. . . . 165
4.4 Biplot quality of the two-dimensional PCA plot for different sides (Figures 4.16a - 4.16b) . . . 166
4.5 PCA Biplot Quality . . . 168
4.6 Axis Predictivity . . . 168
4.7 PCA loadings for first five principal components . . . 168
4.8 CVA Biplot Quality in Original Variables . . . 189
4.9 Axes Predictivity Values . . . 190
4.10 Input data for GPI indices . . . 196
4.11 Coincident table results for one week for process performing within expectation for the integrated versus fundamental GPI . . . 196
4.12 Mean and standard deviation values for scaled reference set $\boldsymbol{Z}$ . . . 196

5.1 Current implemented R packages . . . 206

Listings

5.1 Table TagStatus `CREATE` statement . . . 209
5.2 Table TagName `CREATE` statement . . . 212
5.3 Table DataStructure `CREATE` statement . . . 218
5.4 Table exceltrawlertbls `CREATE` statement . . . 226
5.5 `gridSVG` example . . . 230
5.6 PCA biplot demonstration . . . 243
5.7 CVA biplot demonstration . . . 244
5.8 PhP R and MySQL Interface Example . . . 247
A.1 `getpidata` C function . . . 269
A.2 `imppidata` C function . . . 273
A.3 `sslpiutils::imppidata` function . . . 276
A.4 `getcurpival` C function . . . 277
A.5 `impcurpival` C function . . . 279
A.6 `sslpiutils::impcurpival` function . . . 281
A.7 `sslpiutils::buildtags` function . . . 281
A.8 `sslpiutils::inittag` function . . . 284
A.9 `sslpiutils::sqltemplate` data . . . 286
A.10 `sslgpipi::updgpidb` function . . . 286
A.11 `ssldtutils::dateconvrt` function . . . 288
A.12 `ssldtutils::datestrip` function . . . 288
A.13 `ssldtutils::exceldateconvrt` function . . . 289
A.14 `ssldtutils::posixdatestrip` function . . . 289
A.15 `ssldtutils::getcurrtime` function . . . 290
A.16 `ssldtutils::comparedates` function . . . 290
A.17 `ssldtutils::daysbetween` function . . . 291
A.18 `ssldtutils::nextday` function . . . 291
A.19 `ssldtutils::prevday` function . . . 292
A.20 `ssldtutils::roundday` function . . . 292
A.21 `ssldtutils::roundhour` function . . . 293
A.22 `ssldtutils::roundnmin` function . . . 293
A.23 `ssldbutils::dbconn` function . . . 294
A.24 `ssldbutils::exptodb` function . . . 295
A.25 `DBExp` C function . . . 297
A.26 `ssldbutils::exptodbc` function . . . 299

A.27 `ssldbutils::errlog` function 300
A.28 `ssldbutils::getdbtables` function 301
A.29 `ssldbutils::droptables` function 303
A.30 `ssldbutils::tblexist` function 303
A.31 `ssldbutils::tblrows` function 304
A.32 `ssldbutils::getallmax` function 305
A.33 `ssldbutils::getmaxmax` function 305
A.34 `ssldbutils::getminmax` function 306
A.35 `ssldbutils::getmaxtime` function 307
A.36 `ssldbutils::getmintime` function 307
A.37 `ssldcsutils::retrdscdata` function 309
A.38 `TimeWeigtedStats` C function 312
A.39 `ssldcsutils::twstats` function 325
A.40 `ssldcsutils:getdata` function 326
A.41 `ssldcsutils:retrexpcalc` function 327
A.42 `ssldcsutils:retrcalcs` function 328
A.43 `ssldcsutils:getsubcalc` function 329
A.44 `ssldcsutils:parsetags` function 329
A.45 `ssldcsutils:dataids` function 330
A.46 `ssldcsutils:dataidsfromcalc` function 331
A.47 `ssldcsutils:getcacheddata` function 331
A.48 `ssldcsutils:cacheutilities` function 334
A.49 `ssldcsutils:cachemaintenance` function 335
A.50 `ssldcsutils:parsecalcs` function 336
A.51 `ssldcsutils:createdatadescr` function 338
A.52 `ssldcsutils:datacrawler` function 338
A.53 `ssldcsutils:tagexist` function 339
A.54 `sslxlutils::trawlxls4data` function 340
A.55 `sslxlutils::xlstrawlworker` function 342
A.56 `sslxlutils::createtagdatafromxls` function 344
B.1 `mltv::mltvcva` function 346
B.2 `mltv::mltvpca` function 351
B.3 `mltv::project` function 354
B.4 `mltv::project.mltv-pca` function 354
B.5 `mltv::axispredictivity` function 356
B.6 `mltv::axispredictivity.mltv-pca` function 356
B.7 `mltv::axispredictivity.mltv-cva` function 357
B.8 `mltv::axismspe` function 358
B.9 `mltv::axismspe.mltv-pca` function 358
B.10 `mltv::axismspe.mltv-cva` function 359
B.11 `mltv::axisinclude` function 360
B.12 `mltv::axisinclude.axispred-predictivity` function 360
B.13 `mltv::axisinclude.axispred-mspe` function 360
B.14 `mltv::createbags` function 361

B.15 `mltv::alphabag` function 361
B.16 `mltv::createconcellipse` function 362
B.17 `mltv::concellipse` function 363
B.18 `mltv::createconvexhull` function 363
B.19 `mltv::convexhull` function 364
B.20 `mltv::calcgowaxes` function 365
B.21 `mltv::axmarkerseq` function 367
B.22 `mltv::createmltvtheme` function 369
B.23 `mltv::biplotguicolors` function 370
B.24 `mltv::ggplotcolors` function 370
B.25 `mltv::pchfunc` function 370
B.26 `mltv::calcextvals` function 371
B.27 `mltv::gmembermat` function 372
B.28 `mltv::maxstring` function 373
B.29 `mltv::getaxlabels` function 373
B.30 `mltv::mltvbipl` function 375
B.31 `mltv::mltvbipl.mltv` function 375
B.32 `GOPA` functions 380

Chapter 1

Introduction

Sasol, South Africa, gasifies bituminous coal to synthesis gas, which is converted to fuels and chemicals via Sasol's suite of hydrocarbon processes. A total of 84 Sasol FBDB™ gasifiers have a combined production of no less than $5 \times 10^6 m^3 n/h$ pure synthesis gas. The Sasol facility in Secunda, South Africa, is the largest coal to syngas production facility of its kind in the world. Since the Secunda coal-to-liquids facility delivers nearly 29% of the fuel requirements in South Africa, the continuous improvement of the gasification plant is of critical importance to the company. In order to further optimise product yields and to increase throughput, a thorough understanding is required of those process parameters that govern gasifier performance. In addition to the normal operating parameters a major contributor to gasification performance is the quality of the coal feedstock. The coal handling facilities involves a complex system that is collectively called the Coal Value Chain (CVC).

1.1 A Complex Chemical Plant

1.1.1 Coal Production

The main feedstock to the Sasol Synfuels factory is coal delivered from six collieries situated around Secunda. All six the collieries are underground coal mines which are collectively managed by Sasol Mining, and produce approximately 45 million tons of coal annually (Swart, 2005). A high level overview of the system is shown in Figure 1.1 (Mabuza, 2011). Two of the mines are situated on the Western side of the Sasol Synfuels factory and four are situated on the Eastern side. Overland conveyors are used to transfer the coal from the different mines to two identical stockpile facilities at the Western and Eastern factory, respectively. Coal from the mines on the Western side is transported to the Western stockpile facility, and coal from the mines on the Eastern side is transferred to the Eastern stockpile facility. However, coal is also transferred via heavy duty conveyor belts between the Eastern and

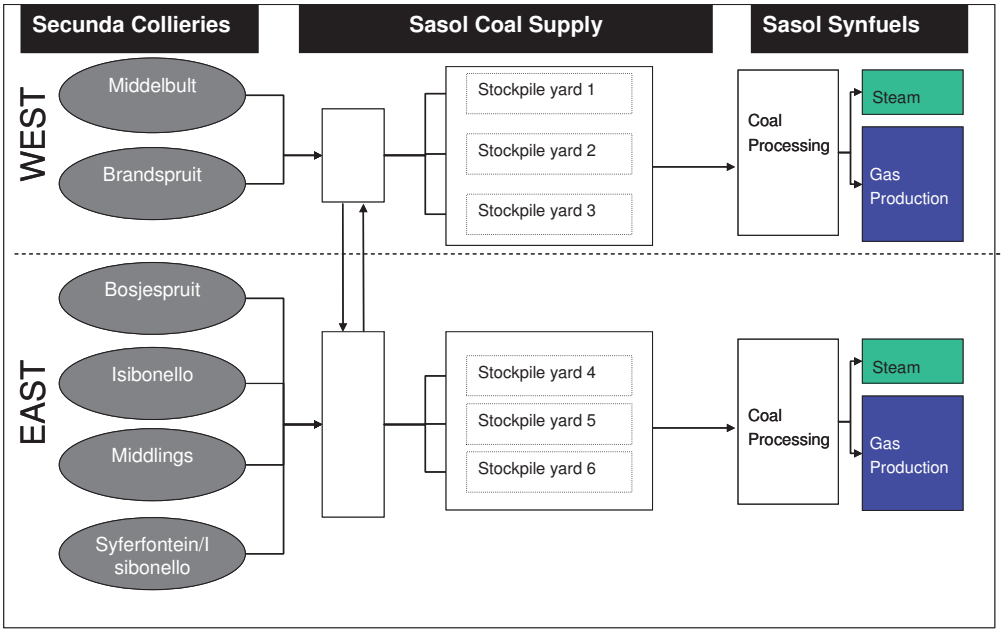


Figure 1.1: Secunda Coal Value Chain (CVC) overview

Western stockpile facilities when required.

Each stockpile facility consists of three stockpile yards. The width of the stockpile is about 600 m, which can contain up to six coal heaps (Mabuza, 2011). The heaps are build by a stacker, and reclaimed by a reclaimer. The stacker and reclaimer operation will be discussed in more detail in Chapter 2. The stockpiles fulfil two distinct roles. First, they act as a supply buffer between the mining operations and the Synfuels gasification operations. Mining is shift based, and coal is not produced continuously. Gasification (and the downstream Synfuels factory) however operates continuously. During mining production excess coal is deposited on the stockpiles, and are then utilised during non-production periods to ensure continuous feed to the gasification process. Second, the stockpiles serve as a homogenisation step. The qualities of the coal vary significantly between the different mines and the run of mine coal from the mines is blended on the stockpiles as it is received. However, great effort is made to reduce the variability in the coal qualities on the heaps through planning and blending. The blending is constrained by the capacity of the stockpile yards and the rate of production at the different mines. A detail discussion of the scheduling and blending problem can be found in Conradie (2007) and Swart (2005).

It is important to note that the current blends are planned and scheduled based on coal qualities that are not available in real time. The main coal

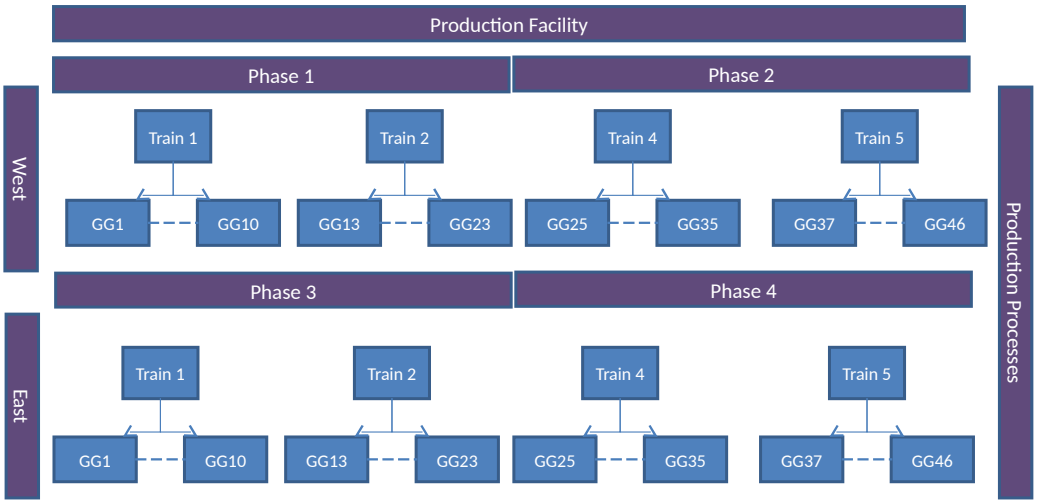


Figure 1.2: Gasification facility overview

property used for blend planning is the ash content of the coal. The laboratory analysis for ash content takes approximately two days, and historic data are therefore used in the blend plan. The turnaround time for the stockpiles is about two to three days, and the implication thereof is that a stockpile could be reclaimed before the quality of the coal on the stockpile is known. To alleviate this, a project was initiated to install instrumentation for on-line measurement of the coal qualities. Specifically, an on-line X-Ray Fluorescence (XRF) analyser was installed to provide real time information on the ash composition (and other elements) of the coal.

1.1.2 Gas Production Facility

The Sasol Coal Gasification (SCG) Plant is a highly complex facility. The system consists of two separate plants that are known as Gasification West and Gasification East and each plant contains four trains, each consisting of 10 or 11 gasifiers (see Figure 1.2). Real time data are captured on more than ten process variables for monitoring the performance of each gasifier, with the main output from the individual gasifiers the amount of raw gas (km^3n/h) produced. All the trains receive the same feedstock, but each train has a different operator. Therefore, differences between trains and production processes may be due to operator, mechanical degradation or other process variable deviations.

An important consideration in this specific production facility is stable operation. The raw gas produced by the gasifiers is the feedstock to the downstream units of the factory. Therefore, it is very important that the raw gas production is as stable as possible, as any upset will result in a disturbance downstream. More detail will be provided in Section 1.2.3.

1.2 Problem Setting

1.2.1 Introduction

The focus of this study is the Sasol Coal Value Chain (CVC) depicted in Figure 1.1. This is a highly integrated system starting from the run of mine coal at the six collieries, the coal blending and reclaiming on the six stockpile yards, up to the 84 gasifiers which convert the coal into raw gas.

The quality of the coal going to gasification has a noticeable effect on gasification performance. Previous studies indicated that the most influential coal qualities are the coal particle size distribution (PSD), and the ash content of the coal as well as the stability of these coal qualities (Coetzer and Keyser, 2002, 2004; Coetzer *et al.*, 2008). Stability in coal feed is one of the main key performance indicators for Sasol Coal Supply.

To achieve optimal coal blends and stable gasification performance, access to real time information is necessary. However, although data are collected across the Coal Value Chain it is stored in different formats and platforms. The integration of the different formats of data into one standardised and centralised data repository is a central component of this study. Clean, standardised and centralised data are the basis on which Data Science is build. In most industrial statistical analysis up to 80% of time is spend on the creation of clean data sets (Wickham, 2005, 2014*b*). The different data sources include the data captured on spreadsheets, distributed control systems (DCS), spectra, and other databases, as well as information captured in fundamental and empirical models.

Figure 1.3 provides a high level overview of the objectives of this study. The main objective of this study is to provide an integrated process monitoring framework for the entire Coal Value Chain. A real time multivariate process evaluation and monitoring methodology is required which possesses the following characteristics:

1. *Data Standardisation* - Data from different sources and formats must be integrated seamlessly into a standardised format.

2. *Real Time Processing* - The data, statistical models and graphical visualisation must be updated in real time.

3. *Deviation Detection* - Deviation from expected performance must be detected, and possible contributing factors highlighted.

4. *Visual Appeal* - The user should be able to visualise with one glance what the state of the plant is.

The various elements depicted in Figure 1.3 have different input data, and have different objectives which need to be addressed. However, there is a cascading of information from top to bottom in the diagram as the performance

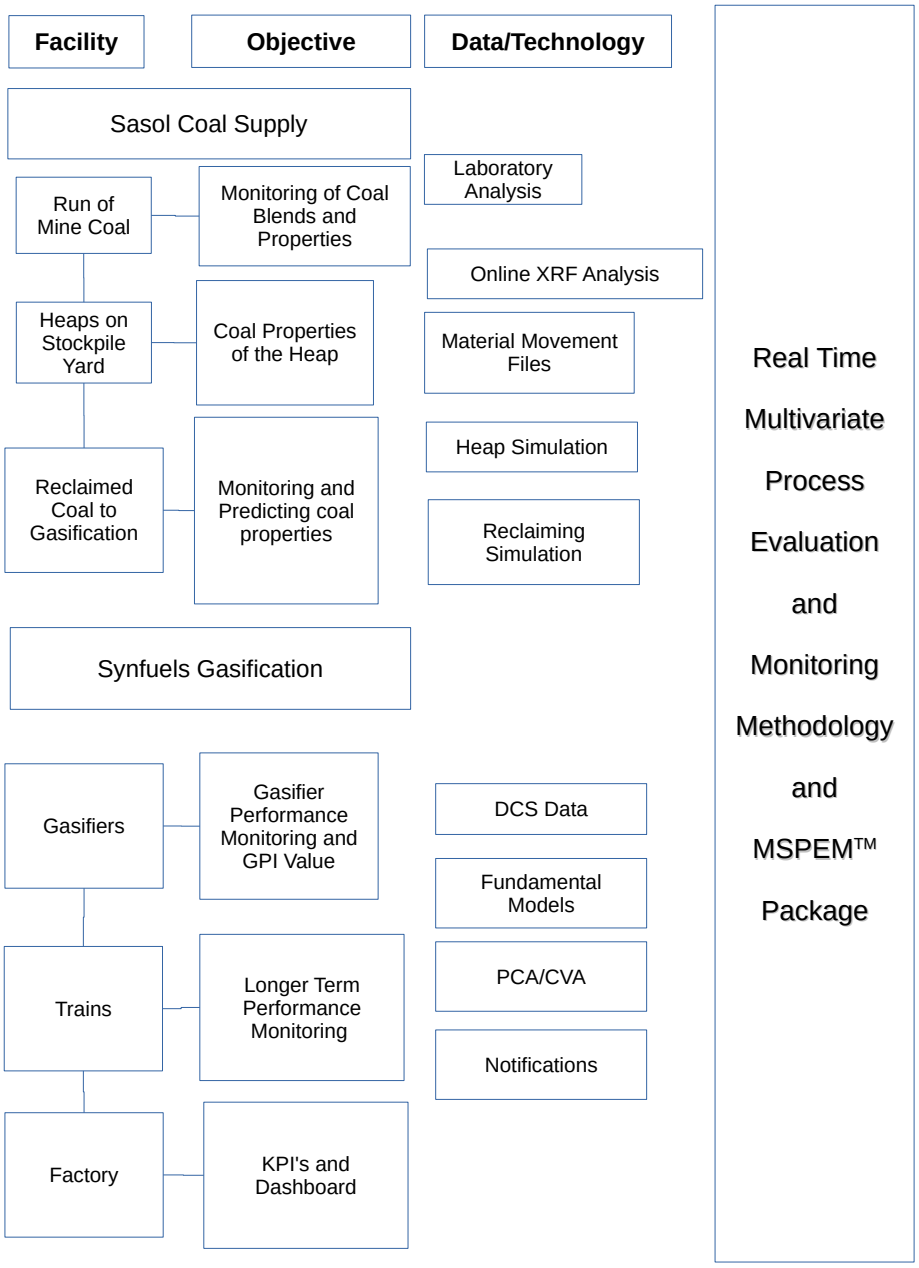


Figure 1.3: Research objectives overview

of each process unit is dependent on the output of the unit preceding it. To address the performance evaluation and monitoring requirements of the CVC many different aspects need to be considered. First, and arguably most important aspect, is whether each unit is operating within expected performance. This calls for process monitoring over a relatively short time interval. A second aspect is how to assess the performance over a longer time period. This information is required to identify equipment failure or process drift.

1.2.2 Coal Value Chain (CVC)

Consider Figure 1.3. Each objective will now be discussed in brief. The coal qualities are different between various mines, and the coal qualities may vary over time for individual mines due to different seams being mined. The coal removed from the mines is blended on the conveyors, but it is possible for a specific seam to go into a specific stockpile. The blend plan at Sasol Coal Supply (SCS) is dependent on the coal qualities of the different mines, more specifically the ash percentage. There is therefore a need to monitor the coal quality of each mine going to the stockpile. Currently, coal samples are taken for laboratory analysis every four hours. However, the laboratory takes two days to perform the ash analysis, and the information is therefore not available in real time. As a potential solution for the delayed ash analysis an XRF coal analyser is currently being tested on one of the stackers at the stack yard for on-line ash (and other ash elements) analysis. In addition, real time information on the coal tonnages going to the stack yard is also available in the SCS database. This data are referred to as the material movement data. In this study the XRF data in combination with the material movement data are used to illustrate real time monitoring of the coal qualities. Furthermore, the laboratory analyses are used to monitor the XRF movements to ensure that the calibration remains stable.

Monitoring the quality of coal on the stack yard is essential for stable feed to the gasification plant. It has been shown previously that coal qualities and stability affect gasification performance (Coetzer and Keyser, 2002, 2004; Coetzer *et al.*, 2008). The coal is blended on the heaps in the stack yard with the aim to produce a uniform blend of feed to gasification. As a first step the actual blend percentages of the different mines on the heaps can be monitored. The blend percentages are calculated in real time from the material movement data. Combining the material movement data with the real time information from the XRF analyser enables predicting the coal quality of the heaps. There exist however some challenges that need to be addressed in using the prediction of the coal quality on the heaps, and consequently the predicted coal qualities being fed to gasification.

- Currently only one XRF analyser is installed on stacker 4 on the Eastern side of the factory. Therefore, it is only possible to directly measure the

coal properties going to the stockpiles at stockpile yard 4. Referring to Figure 1.1, all four mines on the Eastern side will be routed to stockpile yard 4. However, coal transfers between the Eastern and Western factory do occur, and coal from the two mines on the Western side will occasionally be routed to stockpile yard 4.

- The XRF analyser provides information on coal qualities (every 90 seconds), and these results need to be integrated with the material movement data in a sensible way to enable making inferences about the coal qualities from different mines.

- An additional challenge that will be addressed in this study is the prediction of coal quality values across the length of the heap. Due to blending irregularities and coal availability constraints the coal heaps may have varying coal qualities across the length of the heap. Knowledge of the coal qualities across the heap can assist in establishing a reclaiming strategy to mitigate the impact of potential instability of gasification.

1.2.3 Coal Gasification

The Sasol Coal Gasification (SCG) complex consist of 84 gasifiers in a configuration of 8 trains split between two separate but identical production facilities (see Figure 1.2). Fixed bed coal gasification reactors are counter-current devices in which a coal bed moves downward by gravity flow through an upward flowing gas stream. Steam and oxygen are fed at the bottom to provide the reactants for the combustion and gasification reactions. The composition and temperature of the product gas and the amount of unburnt carbon in the ash largely determine the thermal efficiency of the process. The product gas composition and temperature depend on the properties of the coal being processed and on operating parameters such as feed rates, feed temperature and reactor pressure. Each of the trains consists of either 10 or 11 gasifiers, operated by a single operator. Each facility is grouped into two phases with two trains per phase. Data are collected in real time on several (more than 10) process variables for each gasifier. The monitoring of SCG should be a stratified approach with the ability to drill down from top to bottom i.e., from a high level dashboard of the total Secunda factory to the two separate gasification facilities down to the phases, trains and ultimately the individual gasifiers.

For the individual gasifiers an efficient multivariate process monitoring methodology is required for those process variables that govern gasifier performance. These variables can be divided into production, utility and stability variables. From a monitoring perspective there are two different but complimentary strategies that can be followed for process monitoring. The first strategy is a process driven approach where information from the subject matter experts is utilised to specify for example the optimal operating ranges for

the variables. Alternatively a data driven approach can be followed where historical data are used to specify the optimal operating ranges. These two approaches do overlap, and in this study an integrated approach will be developed for the monitoring strategy. Consider Figure 1.3; monitoring of the gasifier process variables is the primary objective. The process data are captured in real time on a distributed control system (DCS), and may be captured at different time intervals for the different variables. From a data perspective some of the problems that need to be addressed entail the selection of an appropriate aggregation window, as well as an appropriate aggregation method for each process variable.

For the individual gasifiers a need was identified for a real time performance index. This gasifier performance index (GPI) should provide a single value which indicates the current health of the gasifier. Three different approaches will be investigated in this study:

- An index developed by process engineers consisting of a weighted deviation from a recommended operating point for each process variable.
- A purely data driven approach will be employed to develop a multivariate performance index making use of historical data collected during periods of good performance.
- The integration of these approaches will be investigated to develop a gasifier performance index which uses both the data and multivariate statistical methodology, as well as the knowledge from subject matter experts.

As all the gasifiers on one train receive the same coal, and are managed by the same operator it is expected that the performance should be similar. Any deviation of performance of a gasifier from the mean performance of all the gasifiers on the train could be an indication of mechanical problems. The objective for the monitoring of the trains is therefore to evaluate the differences between the gasifiers on a train. This is a longer term approach, but should still be available in real time. The ability of Canonical Variate Analysis (CVA) biplots will be investigated to monitor the gasifier differences.

1.2.4 Summary of Research Objectives

In summary, the goal of this study is to develop an integrated monitoring methodology for the Sasol Coal Value Chain. The methodology must be real time, visual, detect deviations from expected performance and also provide for the integration and standardisation of data from different data sources and formats. A high level overview of the objectives is provided in Figure 1.3.

Real time coal quality analysis from the XRF analyser will be summarised and integrated with the material movement files from Sasol Coal Supply, as

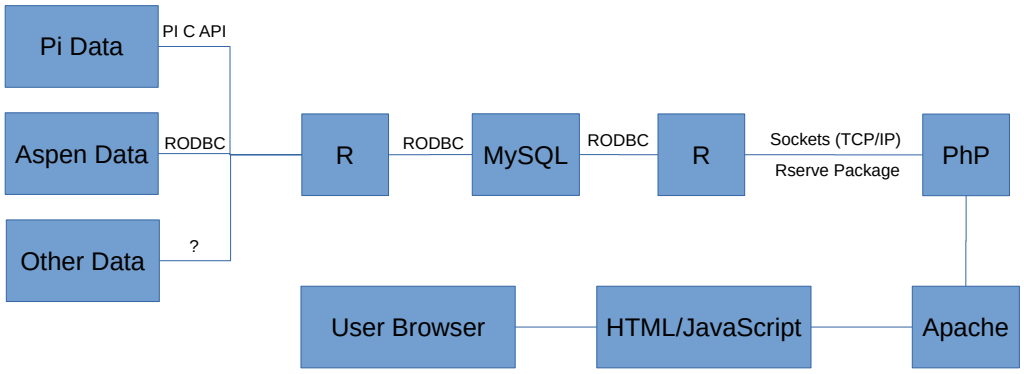


Figure 1.4: Software infrastructure overview

well as various other data sources, to provide information on the coal quality of each mine in real-time. In addition, information on the coal quality of each heap and the quality of the reclaimed coal going to gasification will be generated by simulations.

Monitoring of coal gasification will be divided into separate levels for gasifiers, trains, and facilities. At the gasifier level, a gasifier performance index (GPI) will be implemented. The selection of a reference set for gasification will be investigated, as well as the integration of process knowledge and a data driven approach to monitoring. On the train and facility level, the comparison of gasifiers and trains will be implemented via a multivariate statistical analysis methodology.

In conclusion, this study will present the development and implementation of a real time, integrated and standardised approach to multivariate process monitoring of the Sasol Secunda Coal Value Chain.

1.3 Research Objectives Breakdown

The main objective of this study is to develop a novel integrated Multivariate Statistical Process Evaluation and Monitoring (MSPEM™) methodology for the Coal Value Chain and gasification process. This methodology should be capable of monitoring multiple but similar units or reactors, and visually provide real-time information and quantification on the individual components of the process, the feed and products, as well as on the overall value chain. A brief overview of the methodology is provided in Figure 1.3. In addition to the overall MSPEM™ methodology some specific problems needs to be addressed that will be discussed in more detail in the following sections.

1.3.1 Software Infrastructure

Due to the real-time nature of the MSPEM™ methodology, it must be able to function without manual intervention. The software performing the data updates, creating statistical models and the advanced graphics must therefore be of such a nature that it is able to function autonomously, in real-time, and flag any exceptions for developer intervention automatically. This would enable the use of the methodology on multiple units without unrealistic demands on the developers. An overview of the software infrastructure is depicted in Figure 1.4. A combination of different technologies is utilised to ensure maximum efficiency, flexibility and maintainability. The technologies are combined in such a manner as to be as modular as possible. The aim is to ensure that the different natural divisions of the software system are as independent of each other as possible, therefore ensuring that the modules can be changed without any adverse effect on the remaining modules. As an example, even though MySQL is currently used as the database system, migrating to a different database system would only entail changing the Open Database Connectivity (ODBC) connection strings. The remainder of the system will be unaffected by the change. Some specific software choices are:

1. *User Interface* Various strategies for user interfaces can be employed. A customised executable module can be developed, spreadsheet software like Microsoft Excel can be utilised, or a web interface can be developed. A web interface yields various advantages, some of which are:

    - *Ease of use* - Most users are familiar with a web interface, and will therefore intuitively know how to navigate a web based application.
    - *Maintainability* - Any software that resides on the client's computer needs to be updated when bugs are fixed, or new versions become available. A web interface ensures that all changes are immediately available to all relevant users as the application resides on a central server. In addition, in any modern information technology landscape, installing custom software on users computers is not allowed. A web based interface ensures that no software will be installed on the users' computers.
    - *Interactive* - Web technologies like JavaScript, SVG, JSON and AJAX make it possible to have highly interactive web based applications.
    - *Smart Devices* - Smart devices like smart phones and tablets are becoming part of the information technology landscape. A web based interface makes the application available on these smart devices without any additional software development. In addition, the core application will be identical for use on any device as long as the development is standard compliant.

2. *Integrated and Standardised Data*

   The availability of integrated and standardised real time data is one of the key underpinnings of any real time monitoring system. The creation of a comprehensive data management framework is one of the key objectives of this study, and is also one of the most time consuming steps in most data driven projects (Wickham, 2014*b*). The Coal Value Chain data reside in different data sources and in different formats that needs to be consolidated and changed into information that can be utilised for decision making.

   The consolidated data are utilised for various modelling exercises, including statistical and fundamental engineering models. These models are used for integrated performance monitoring, performance indices, and performance comparisons. Availability of accessible historical data enables the identification of long term historical trends.

3. *Graphical Visualisation*

   Graphical visualisation is an important aspect of performance evaluation and monitoring, as tables of data are in general difficult to interpret for multivariate data (Few, 2012, 2013). A design goal of MSPEM™ is for the user to be able to evaluate the state of the system at one glance, and be guided to the important information and a problem area quickly. Biplots as a multivariate approach to the visualisation have much to offer, specifically the new biplot philosophy proposed in Gower and Hand (1996) and further elaborated on in Gower *et al.* (2011). Biplots provide an intuitive visual summary of large volumes of information that would be challenging to extract from tables of data or univariate scatter plots. Biplots combine the information contained in both the columns and rows of the data. Combined with tools like quality ellipses and alpha bags, the biplot methodology provides information on the correlation between variables, the trends over time, unexpected behaviour, and the variables that contribute to the unexpected behaviour (Aldrich *et al.*, 2004; Gower *et al.*, 2011; Sparks *et al.*, 1997).

   However, there are some challenges to the application of biplots for real time process monitoring:

   - *Reference Data.* A reference data set is necessary if methods like quality ellipses are to be used to monitor the system for unexpected behaviour. Choosing the reference set however is difficult, especially for a process with multiple units such as the gasification process. It is advantageous to use a single reference set for all gasifiers, as this makes comparison of performance sensible. However, this is not always practical as different operating philosophies, different

coal qualities, and measurement errors may lead to shifts in the operating region.

- *Performance index.* For effective performance monitoring of a process with multiple units (see Figure 1.2) an aggregated performance index is required for summarising performance deviations. This should preferably be a statistic that is comparable over the different units, and it must be possible to aggregate to higher level views i.e., trains and plant/facility. The performance index should be comparable to the results obtained from the biplot theory.
- *Predictivities* A natural consequence of the projection of a higher dimensional space onto a lower dimension is that some information is lost in both the samples and the axes representation (Gardner-Lubbe *et al.*, 2008). Generally this is not a concern as the analyst can inspect the measures of fit. Using the biplot as monitoring device without indicating to the casual user the measures of fit can be misleading, as users are accustomed to scatter plots where the variables and samples are represented perfectly. This can lead the user to make invalid conclusions from the plots.

1.3.2 Fundamental Computer Models

Computer models are becoming an integral part of process design and decision making in industry (Fang *et al.*, 2006; Lin *et al.*, 2001; Santner *et al.*, 2003). These models are generally highly non-linear with a large number of input and output variables, of which some of the relationships are not necessarily known. In addition, the model outputs are deterministic i.e., a given set of input parameters will always map to the same set of outputs. This necessitates a design that allows for a wide range of models. This is in contrast to stochastic simulation models which are more akin to physical experiments as each replication will produce a different result (Rossouw *et al.*, 2010). The recommended design strategy is to employ space-filling designs for running the computer code (Fang *et al.*, 2006, 2000; Lin *et al.*, 2001; Sacks *et al.*, 1989). In the petrochemical industry the inputs to computer models are commonly a mixture of chemical components or molecules. Finding a space filling design for constrained mixture experiments is a challenging problem (Borkowski and Piepel, 2009). In the multidimensional scaling literature it is well known that the choice of a dissimilarity metric is an important consideration (Cox and Cox, 2001). In this study the choice of a dissimilarity metric for space filling designs for compositional computer experiments will be discussed and applied (Coetzer *et al.*, 2012).

1.3.3 Choice of Reference Set

A very important aspect which needs to be addressed in any process evaluation and monitoring methodology is to be able to detect whether the process is deviating from expected performance. Therefore, the expected behaviour of the process needs to be defined. This is normally achieved by selecting a reference set of data from a historical period of time where the process was running stable and within expectation. The correct selection of the reference data is crucial to the success of process monitoring.

Defining a criterion for selecting the reference set, and determining the number of points to select are dependent on the statistic that is used to define deviations from expected performance. The reference set should be chosen such that the average run length (ALR), that is the average length of time without signalling a deviation is acceptable i.e., the statistic is not over sensitive, and frequently signal deviations, and it is not too insensitive and miss real deviations (Russell *et al.*, 2000). An additional aspect of choosing a reference set is detecting when the deviation is not just a short term performance deviation, but a long term process drift due to feedstock changes. In the case of a process drift, the reference set should be updated to reflect expected behaviour in the new operating region.

Some important aspects concerning Reference Data are:

1. The criterion for choosing the historical period for specifying the reference data.

2. The number of data points for the reference set i.e., a balance between instability versus over fitting.

3. When should the reference set be updated.

Generalised Orthogonal Procrustes Analysis (GOPA) (Gower and Dijksterhuis, 2004) will be investigated in this study for selecting the optimal reference set for the multivariate monitoring of the multiple identical production processes (Coetzer *et al.*, 2014).

1.3.4 Comparison of Performance

In the petrochemical industry it is of interest to compare the performance of different production units. The Sasol coal gasifiers provide an interesting performance comparison opportunity as the reactors are identical, and receive similar feed. As there is one operator per train of gasifiers, a gasifier with different performance characteristics compared to the remaining gasifiers on the train can indicate either mechanical failure or measurement errors. The number of variables, and the interrelationships between these variables, suggests a multivariate approach to the comparison of the gasifiers. Canonical Variate Analysis (CVA) biplots is an approach that could be beneficial. CVA biplots

provide a graphical visualisation of the multivariate differences between the variable means for the groups. The axes provide an indication of the variable contributions to the differences (Gower *et al.*, 2011; Gower and Hand, 1996).

The CVA biplot is however a two dimensional approximation of a multi-dimensional space. This has implications for both the representation of the samples, as well as the variables (axes) (Gardner-Lubbe *et al.*, 2008). In an offline analysis the predictivities of the axes can be inspected, and removed if they are not represented well in the current two dimensional space. Different combinations of eigenvalues can be evaluated to find a space where the overall quality of the CVA biplot is acceptable, as well as the predictivity of the axes. In a real time application this process must however be automated to ensure the user does not draw erroneous conclusions from the CVA biplot. The real time evaluation of CVA axes predictivities, as well as eigenvalue combinations will be addressed in this study.

1.4 Thesis Outline

The rest of the thesis is outlined as follows:

- Chapter 2 - The integration of various diverse data sources in combination with simulation models to generate real time coal quality information on the coal heaps as well as the reclaimed coal are discussed.
- Chapter 3 - The development of a multivariate, real-time, data-driven approach to the monitoring of the Sasol Coal Gasification plant is presented.
- Chapter 4 - The development of a fundamental, empirical, and finally integrated gasifier performance index (GPI) is discussed.
- Chapter 5 - The software infrastructure developed to implement the real-time MSPEM™ application is presented.
- Chapter 6 - The conclusions of this study are summarised, and some topics for future research presented.

Supplementary to the main body of the thesis three appendices are provided. These appendices contain details of the computer code that was written to develop the MSPEM™ application as well as illustrations of the functioning of the implemented user interface:

- Appendix A - The code listings for the data interface software.
- Appendix B - The code listings for the multivariate graphical software.
- Appendix C - Illustrative screen grabs of the functioning of the implemented user interface.

1.5 Acronyms

Table 1.1: List of acronyms used in this thesis

API	Application Programming Interface
CSRP	Coal Stacking and Reclaiming Picture
CVA	Canonical Variate Analysis
CVC	Coal Value Chain
DCS	Distributed Control System
GG	Gasifier
GOPA	Generalised Orthogonal Procrustes Analysis
GPI	Gasifier Performance Index
MDS	Multidimensional Scaling
MSPE	Mean Standard Predictive Error
MSPEM	Multivariate Statistical Process Evaluation and Monitoring
ODBC	Open Database Connectivity
PC	Principal Component
PCA	Principal Component Analysis
PSD	Particle Size Distribution
ROM	Run of Mine
SCS	Sasol Coal Supply
SPC	Statistical Process Control
SQL	Structured Query Language
SVD	Singular Value Decomposition
XRF	X-Ray Fluorescence analyser
m^3n/h	Normal cubic meters per hour

Chapter 2

Sasol Coal Supply

This chapter will be focusing on developing a visual, real-time interface for the integrated Coal Value Chain (CVC). In addition, various statistical and Data Science techniques will be applied to close the gap between the data that are available, and the requirements with regards to real time coal quality information of the CVC. Consider Figure 1.1. This chapter will focus on the Sasol Coal Supply (SCS), starting with the single source coal from the Secunda Collieries, up to and including the reclaimed coal from the stockpile yards transferred to the gasification plant. Coal qualities are the integrating factor between the different facilities of SCS, as the coal is transferred from the mines to gasification. The quality and stability of the coal feed have a notable impact on gasification performance. As discussed in Section 1.2.2 the most influential coal qualities are the coal particle size distributions (PSD), the ash content of the coal as well as the overall stability of these qualities (Coetzer and Keyser, 2002, 2004; Coetzer *et al.*, 2008). Real-time coal quality information across the SCS facility will therefore enable not only predictability of the gasifier performance, but will additionally, and equally important, allow for the implementation of mitigation steps at gasification during periods of non-optimal feed.

Coal qualities have to be tracked through several stages from mining, stacking and reclaiming as depicted in Figure 2.1. At the first stage, the ash content and particle size distribution (PSD) are to some degree influenced by mining practices at the Secunda collieries. Mining practices are not inside the scope of this study and will be treated as a given. The coal transferred on the SCS conveyor system from the collieries is called run of mine (ROM) coal, or single source coal, as no coal blending has taken place at this stage. The coal qualities of the ROM coal are utilized in the weekly SCS Blend Plan, as the blend plan is mostly governed by the ash content and quality of the heaps on the stockpile yard. The actual scheduling of the SCS Blend plant is also outside the scope of this study, and is discussed in detail in Conradie (2007) and Swart (2005).

The blend plan prescribes the stacking process, and given the ROM coal

qualities, it yields the coal quality of the heaps at the stockpile yard. Due to constraints on the conveyors as well as production constraints at the mines, the blend plan is not followed perfectly for every heap, and there can be significant variation in the actual blends between the different heaps. Additionally, the layering of coal is not necessarily homogeneous, and a heap may therefore have varying coal properties across its length. The properties across the length of a heap have a direct impact on the properties of the reclaimed coal. In addition, at least two (but possibly three) heaps are reclaimed simultaneously. The properties of the reclaimed coal are therefore dependent on the following information:

1. The proportions of each mine on the heap and across the length of the heap.

2. The coal properties of each mine on the heap when it was stacked.

3. The combination of heaps reclaimed as well as the tonnages of each heap reclaimed.

The heap stacking and reclaiming process will be discussed in detail in Section 2.3.

An overview of the information required and existing data across the CVC are shown in Figure 2.2. The types of data and data collection will be discussed in detail in Section 2.1. The goal of this chapter is to demonstrate the use of the existing data to create real time information across CVC.

Receiving coal quality information two days after the heaps have been completed limits the value of the information. An additional constraint on the information available is that it is not available in a standardized format. The information is spread over various databases in different formats, spreadsheets distributed via e-mail, as well as different intranet sites. In this chapter the development of a consistent data management framework as well as appropriate multivariate visualization techniques will be developed to create a standardized, real-time, visual tool for CVC.

2.1 Data Sources

One constraint to the efficient use of the available information for SCS is the various formats and locations of the data. This makes it very difficult to get an integrated view of the system. A major part of this project was the development of a consistent and integrated data management framework. An integrated and consistent data framework is a prerequisite for developing a real time monitoring system. The different data sources will now be discussed.

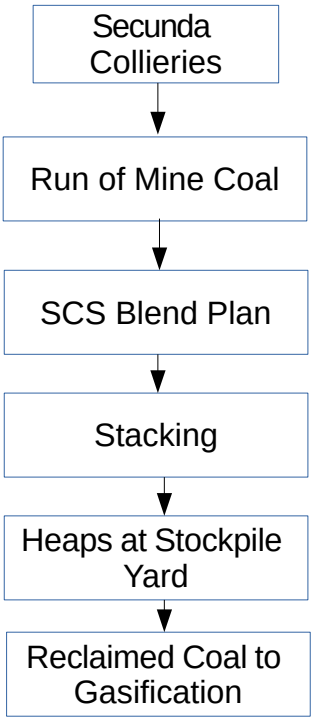


Figure 2.1: Secunda Coal Supply (SCS) overview

2.1.1 Excel Files from SGS Laboratories

Each morning a set of Excel spreadsheets containing laboratory coal analyses including ash percentage, particle size distribution and moisture in the coal is distributed via an e-mail distribution list. The SGS Laboratory analyses are not currently available in any DCS or database at SCS, which necessitated the development of an automatic data capturing strategy. These spreadsheets originate from various mail addresses, and are not sent at the same time each day. In addition, the mails are not distributed on Sundays. In general the spreadsheets contain the respective coal quality analyses for the current month. The names of the spreadsheets, and the worksheets in the spreadsheets change periodically. The format of the sheets is however consistent. It is therefore possible to capture the information in the spreadsheets automatically by triggering visual basic for applications (VBA) macros when the mails arrive in the Microsoft Outlook inbox.

2.1.2 XRF Data

The XRF analyzer will be discussed in detail in Section 2.2. One XRF analyzer is installed at Stacker 4 on the coal stack yard. The data from the analyzer are captured on a local PC by the Monaco software from Springer Technology. The data are stored in a local MySQL database. The data of interest to this study

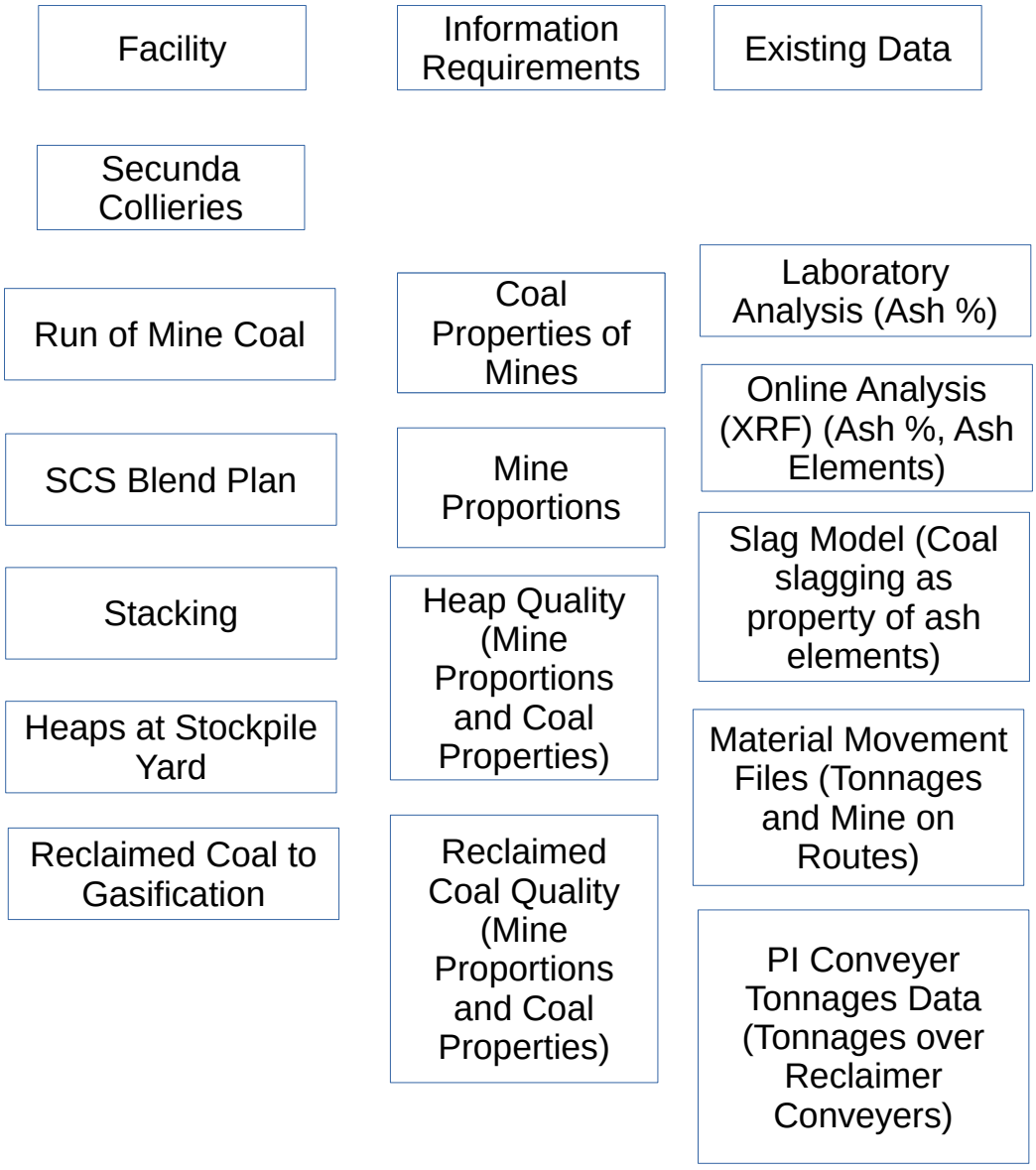


Figure 2.2: Secunda Coal Supply (SCS) data overview

consist of a table called Spectra, which specifies numbers and time stamps for each spectra. An extract of such a table is shown in Figure 2.3. Figure 2.4 shows a table called Results, which consists of spectra numbers, element IDs and values. The element IDs are mapped to elements in a separate element table. The data are filtered to some extent in that a lower limit is set on the X-Ray count rate. This is to exclude data where very small volumes of coal pass under the XRF sled, or when the belt is empty. It is important to note that the XRF has no intelligence of mines, and treats all coal data exactly the same. Note that XRF data in isolation are not useful, and does not provide the relevant information regarding the individual mines, or the properties of the coal on the heaps.

2.1.3 Material Movement Data

The material movement information is generated in the SCS control room when the different mines and tonnages are selected for transport on the conveyors, and captured in a Microsoft SQL Server database. The material movement data record the movement of coal on the conveyors from the different mines to the stackers, and from the different heaps reclaimed to gasification. This information is invaluable, since in combination with the XRF data it can be used to build a profile for each mine. In addition, this information can be used to predict the properties of the coal for each individual heap. This will be discussed in more detail in Section 2.3. The material movement file contains information on the mine from which the coal originates, the unique ID of the heap the coal is sent to, the begin and end times of the transfer, and the total tons of coal transferred in the specified time range. Figure 2.5 depicts a screen grab of the material movement file in Excel. Note that due to confidentiality constraints the actual mine names were encoded as Mine A to Mine E. This file contains the material movement for all six stackers. Figure 2.6 shows a SCS material movement file filtered for a specific heap viz Heap ID 24678.

The data from the SQL Server database are downloaded in the R software (R Core Team, 2015), and the dates are converted to the standard format discussed in Section 5.2. The converted data are then exported to a local MySQL database for convenient and efficient access. Figure 2.7 depicts a screen grab of a converted material movement file in the MySQL database.

2.1.4 Stockpile Information Files

Information about the stockpiles is made available on Excel files two times a day (06:00 and 18:00) by the mining department. These files contain various information, for example the length of the heap (in meters), start and end position, planned tons and actual tons. The start and end positions are defined on a scale of meters from +300 to -300 relative to a zero point in the centre of the stack yard. The positive values are on the factory side of the stack yard,

ID	TimeStart
504853	1403608938
504854	1403609034
504855	1403609131
504856	1403609228
504857	1403609325
504858	1403609422
504859	1403609519
504860	1403609616
504861	1403609713
504862	1403609810
504863	1403609907
504864	1403610005
504865	1403610102
504866	1403610199
504867	1403610296
504868	1403610393
504869	1403610490
504870	1403610586
504871	1403610683
504872	1403610780
504873	1403610876
504874	1403610972
504875	1403611069

Figure 2.3: XRF Spectra table

SpectrumID	ElementID	Value
410612	103	21.5768
410612	102	2.35738
410612	101	-0.720933
410612	100	21.0344
410612	22	4.12617
410612	14	1.51882
410612	10	3.83186
410612	8	0.0151...
410612	5	1432.14
410612	4	4.65786
410612	3	59.4543
410612	2	23.4665
410612	1	2.6458
410612	0	29.8387
410611	132	30.4839
410611	124	6353.04
410611	105	46.8524
410611	104	21.6037
410611	103	20.7785
410611	102	1.79503
410611	101	-0.796126
410611	100	20.2584
410611	22	3.43176

Figure 2.4: XRF Results table

HEAP_ID	PRODUCT	START_DATE	END_DATE	WEIGHT	FLAG	NODE_NAME	SCS_ID
25152	Mine B	2014/06/24 09:38	2014/06/24 11:37	2933.46	1	HB3143	15154
25153	Mine A	2014/06/24 09:39	2014/06/24 12:19	3492.72	1	HB5030	15155
25150	Mine B	2014/06/24 11:47	2014/06/24 13:10	2063.14	1	HB2163	15153
25152	Mine C	2014/06/24 11:57	2014/06/24 13:47	2653.84	1	HB3030	15154
25155	Mine D	2014/06/24 12:15	2014/06/24 13:06	1091.25	1	HB1000	15157
25153	Mine A	2014/06/24 12:25	2014/06/24 13:04	938.84	1	HB5030	15155
25155	Mine D	2014/06/24 13:09	2014/06/24 13:13	103.79	1	HB1000	15157
25155	Mine D	2014/06/24 13:13	2014/06/24 13:13	76.63	1	HB1000	15157
25155	Mine D	2014/06/24 13:13	2014/06/24 13:21	228.92	1	HB1000	15157
25149	Mine A	2014/06/24 13:14	2014/06/24 14:05	1280.86	1	HB6030	15152
25150	Mine B	2014/06/24 13:16	2014/06/24 14:09	1308.06	1	HB2163	15153
25155	Mine D	2014/06/24 13:22	2014/06/24 13:40	537.38	1	HB1000	15157
25154	Mine B	2014/06/24 13:22	2014/06/24 14:09	1229.8	1	HB4103	15156
25155	Mine D	2014/06/24 13:41	2014/06/24 14:27	1293.01	1	HB1000	15157
25152	Mine C	2014/06/24 13:50	2014/06/24 13:54	37.24	1	HB3030	15154
25152	Mine C	2014/06/24 13:57	2014/06/24 15:32	2842	1	HB3030	15154
25149	Mine A	2014/06/24 14:08	2014/06/24 14:12	38.22	1	HB6030	15152

Figure 2.5: Sasol Coal Supply (SCS) Material Movement file

HEAP_ID	PRODUCT	START_DATE	END_DATE	WEIGHT	FLAG	NODE_NAME	SCS_ID
24678	Mine B	2014/03/07 14:27	2014/03/07 14:29	24.94	1	HB4103	14755
24678	Mine B	2014/03/07 14:38	2014/03/07 17:37	4308.6	1	HB4103	14755
24678	Mine D	2014/03/08 19:40	2014/03/08 19:59	426.8	1	HB4000	14755
24678	Mine C	2014/03/08 22:07	2014/03/08 23:30	2194.22	1	HB4010	14755
24678	Mine D	2014/03/09 21:53	2014/03/09 23:51	2877.02	1	HB4000	14755
24678	Mine D	2014/03/10 00:06	2014/03/10 00:17	337.56	1	HB4000	14755
24678	Mine D	2014/03/10 01:20	2014/03/10 02:03	889.49	1	HB4000	14755
24678	Mine A	2014/03/10 02:39	2014/03/10 02:45	39.2	1	HB4020	14755
24678	Mine A	2014/03/10 02:52	2014/03/10 03:37	581.14	1	HB4020	14755
24678	Mine B	2014/03/10 03:41	2014/03/10 03:42	12.04	1	HB4103	14755
24678	Mine B	2014/03/10 03:44	2014/03/10 03:45	24.94	1	HB4103	14755
24678	Mine B	2014/03/10 03:47	2014/03/10 03:48	14.62	1	HB4103	14755
24678	Mine B	2014/03/10 03:54	2014/03/10 04:27	749.06	1	HB4103	14755
24678	Mine E	2014/03/10 04:36	2014/03/10 05:21	639.94	1	HB4080	14755
24678	Mine B	2014/03/10 05:18	2014/03/10 05:39	447.2	1	HB4103	14755
24678	Mine B	2014/03/10 05:57	2014/03/10 06:01	94.6	1	HB4103	14755
24678	Mine E	2014/03/10 06:02	2014/03/10 06:51	954.52	1	HB4080	14755
24678	Mine B	2014/03/10 06:51	2014/03/10 07:32	975.24	1	HB4103	14755
24678	Mine B	2014/03/10 12:53	2014/03/10 14:23	2083.78	1	HB4103	14755
24678	Mine B	2014/03/10 16:34	2014/03/10 17:42	1623.68	1	HB4103	14755
24678	Mine B	2014/03/10 18:08	2014/03/10 18:22	309.6	1	HB4103	14755
24678	Mine A	2014/03/10 18:58	2014/03/10 22:39	6914.88	1	HB4020	14755
24678	Mine A	2014/03/10 23:05	2014/03/11 00:49	2724.4	1	HB4020	14755
24678	Mine B	2014/03/11 06:23	2014/03/11 06:23	598	1	HBMAIN	14755
24678	Mine B	2014/03/11 06:30	2014/03/11 06:30	2324	1	HBMAIN	14755

Figure 2.6: Sasol Coal Supply (SCS) Material Movement file for a specific heap ID

ID	HEAP_ID	PRODUCT	START_DATE	END_DATE	WEIGHT	FLAG	NODE_NAME	SCS_ID
131703	23306	Mine B	1365056222	1365058170	778.3	1	HB5103	13467
131704	23306	Mine B	1365065790	1365069990	1597.02	1	HB5103	13467
131705	23311	Mine B	1365067200	1365067799	241.66	1	HB2163	13472
131706	23315	Mine C	1365066783	1365070593	1397.48	1	HB3020	13476
131707	23323	Mine B	1365067920	1365067952	58.48	1	HB2163	13484
131708	23323	Mine D	1365068280	1365069211	129.01	1	HB2020	13484
131709	23319	Mine B	1365070025	1365070110	8.6	1	HB1124	13480
131710	23322	Mine A	1365081480	1365098853	8481.9	1	HB5030	13483
131711	23324	Mine D	1365083340	1365087450	1966.19	1	HB1000	13485
131712	23310	Mine B	1365084960	1365096182	4696.46	1	HB4103	13471
131713	23322	Mine A	1365098912	1365099002	23.52	1	HB5030	13483
131714	23322	Mine A	1365099210	1365106040	3408.44	1	HB5030	13483
131715	23310	Mine D	1365099810	1365106022	3007.97	1	HB4000	13471
131716	23324	Mine C	1365106171	1365106260	48.02	1	HB1010	13485
131717	23325	Mine D	1365111152	1365122101	5291.35	1	HB3010	13486
131718	23324	Mine B	1365111243	1365131702	8963.78	1	HB1124	13485
131719	23318	Mine B	1365111543	1365117900	2634.18	1	HB6103	13479
131720	23322	Mine C	1365116250	1365122192	2564.66	1	HB5010	13483
131721	23326	Mine A	1365118383	1365123482	2477.44	1	HB4030	13487

Figure 2.7: Sasol Coal Supply (SCS) Material Movement file with converted date and time

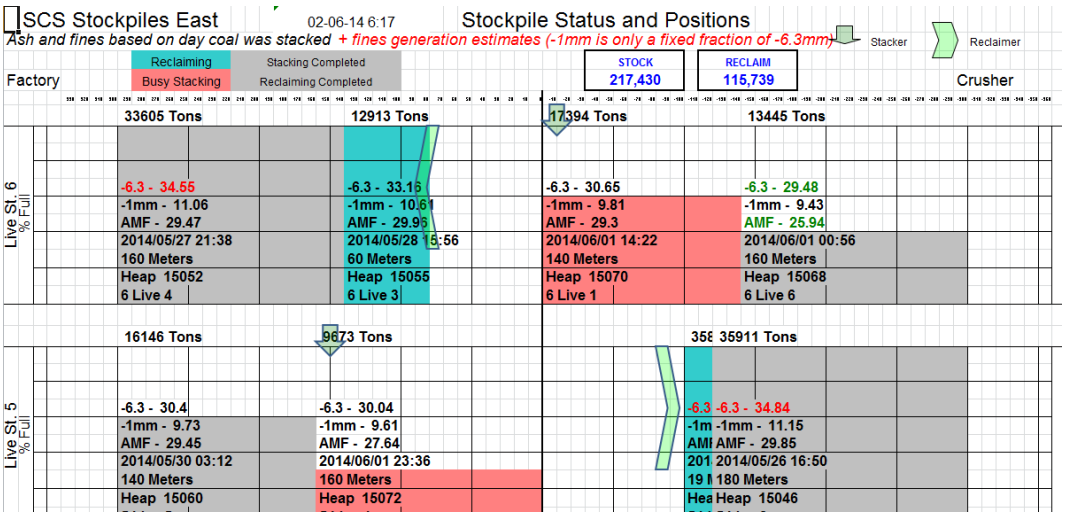


Figure 2.8: Sasol Coal Supply (SCS) Coal Stacking and Reclaiming Picture (CSRP)

and the negative values are on the mining side of the stack yard. Therefore, a heap defined with start point -100 and end point -200 will have a length of 100, and is located on a point starting 100 meters from the centre of the stack yard to the mining side, up to 200 meters from the centre of the stack yard to the mining side (this will be explained in more detail in Section 2.3.3). The information is depicted in a graphic representation. This is referred to as the Coal Stacking and Reclaiming Picture (CSRP). An example is shown in Figure 2.8. The information in these files is not currently available on any other system, and capturing the information on these files programmatically was therefore essential to the CVC project.

To capture these data in a convenient format a combination of R scripts and Excel VBA macros is utilized. An R script was developed that reads the web interface data, and extracts a list of the CSRP files using regular expressions (Friedl, 2006). This list is then compared to a directory listing of the downloaded CSRP files, and the missing files downloaded via the R curl command. Excel VBA macros are then utilized to extract the relevant information and export it to the MySQL database. An extract of the information is shown in Figure 2.9. The information in Figure 2.9 is not the only information contained in these files, and the remaining information will be discussed in more detail later in the chapter.

2.1.5 Data Integration

Although all the information discussed in the previous sections are available, individually it is very difficult to obtain any useful insight from the different sources. Maximum intelligence can only be generated through the combination of all the different sources of data. The Sasol MSPEM™ package captures all

SCS_ID	LENGTH	TONSPLAN	STACK	SIMSTACK	STARTPOS	ENDPOS	RECLDIR
15118	180	36000	6	1	0	180	1
15119	140	28000	3	1	-290	-150	0
15120	50	9250	5	1	-320	-270	0
15121	60	12000	1	1	-200	-140	1
15122	120	22200	5	1	-270	-150	0
15123	140	28000	1	1	-300	-160	0
15124	150	27750	5	1	-150	0	0
15125	180	36000	2	1	0	180	0
15126	200	40000	4	1	100	300	0
15127	150	30000	3	1	-150	0	0
15128	160	32000	1	1	-160	0	0
15129	100	20000	4	1	0	100	0
15130	100	18500	5	1	0	100	0
15131	160	32000	6	1	-300	-140	0
15132	100	20000	3	1	100	200	0
15133	140	28000	6	1	-140	0	0
15134	100	18500	5	1	100	200	0
15135	140	28000	2	1	-140	0	0

Figure 2.9: Sasol Coal Supply (SCS) stockpile information data

the information in a convenient and usable format in one central database discussed in more detail in Section 5.3.

2.2 X-Ray Fluorescence analyzer (XRF)

The theoretical chemistry of the XRF analyzer is outside the scope of this study. However, a short explanation of the measurements from the XRF analyzer will be given here (Moitsheki *et al.*, 2014). In principle, XRF is the emission of characteristic "secondary" (or fluorescent) X-rays from a material that has been excited by bombarding with high-energy X-rays or gamma rays. Primary X-rays are bombarded onto a sample. X-rays are either absorbed by the atom or scattered by the material. During this process, if the primary X-ray had sufficient energy, electrons are ejected from the inner shells, creating vacancies. These vacancies present an unstable condition for the atom. As the atom returns to its stable condition, electrons from the outer shells are transferred to the inner shells and in the process yield a characteristic X-ray whose energy is the difference between the two binding energies of the corresponding shells. Each element has a unique set of energy levels and therefore produces X-rays at a unique set of energies.

One XRF analyzer has been installed on Stacker 4 at the Eastern Stock Yard. The XRF runs on a sled on top of the coal passing on the coal conveyor (See Figure 2.10). This ensures proximity between the X-Ray tube and the

Figure 2.10: X-Ray Fluorescence Analyzer (XRF)

coal on the conveyor. The calibration of the XRF analyser is outside the scope of this study as it is performed by the supplier (Springer Technologies). It is however important to note that the calibration is performed on run-of-mine coal samples which ensures that the results are robust.

2.3 Stacker Simulation Model

The current location of the XRF (on the conveyor going to the stacker as shown in Figure 2.11) brings with it both opportunities and challenges. The greatest challenge from a gasification process perspective is "knowing" (predicting) the properties of the reclaimed coal going to gasification. However, the unique opportunity is to specify a real time profile of the coal qualities from the different mines measured by the XRF analyser. In this section a solution will be presented to this challenge by utilising the real time coal profile information from the XRF in combination with the material movement information from SCS to create a simulation model for the stacking process.

In any simulation study computational efficiency is a concern. Various stacker models are available commercially and in the literature (Cipold, 2013)

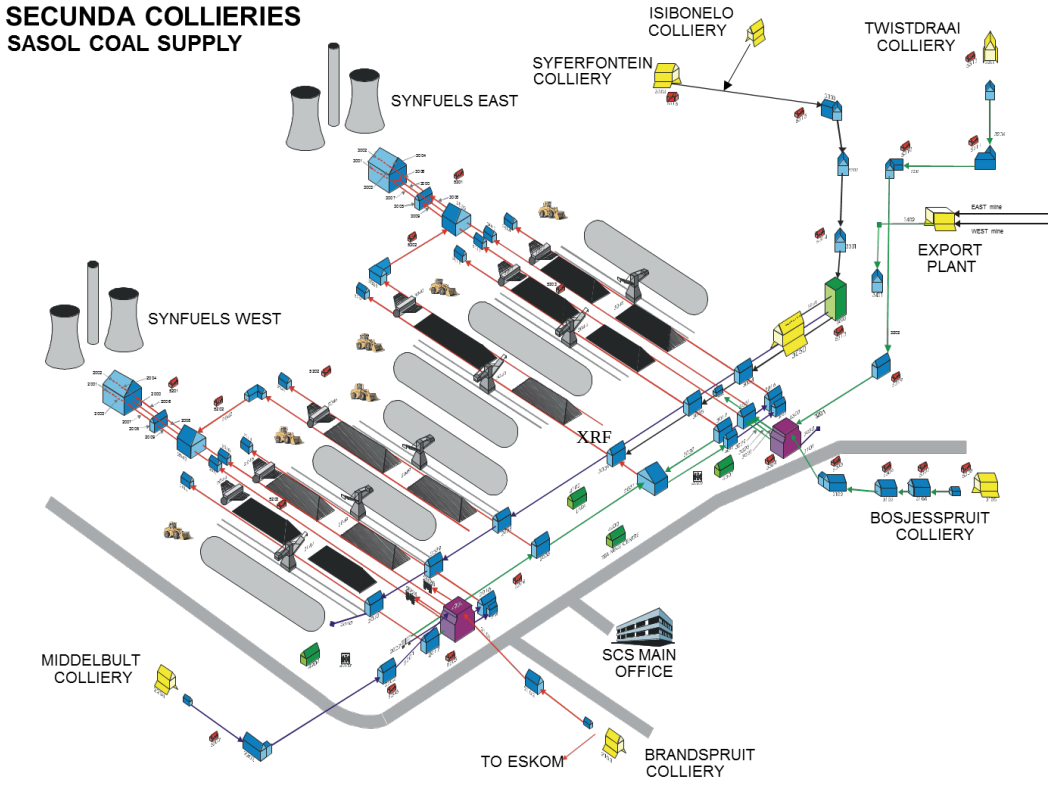


Figure 2.11: Position of XRF on conveyor to Stacker 4

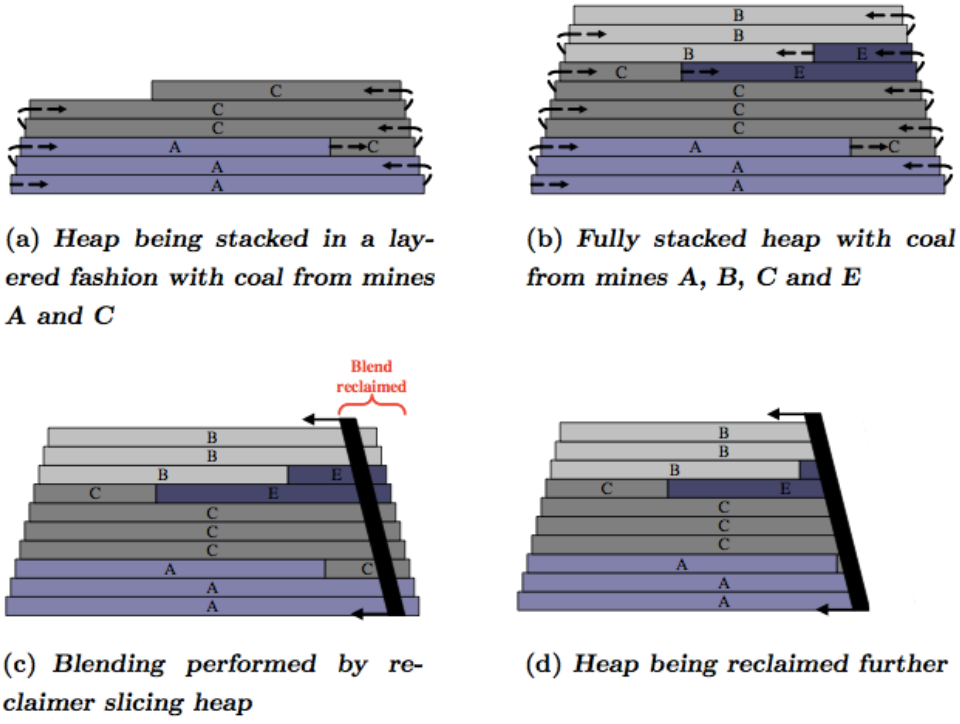
but most of these modeling approaches are not feasible for a real time application as they are computationally very expensive, and can take very long (hours) to converge to a solution. Simulation in general entails a compromise between detail and speed. A more detailed model will be computationally more expensive, and will therefore take longer to converge. A brief overview of the stacking/reclaiming technologies used at the coal stacking yard will now be provided to assist in the understanding of the modeling methodology.

2.3.1 Stacking/Reclaiming Technology Overview

Huge variation in coal quality exists within and between the different mines. To homogenize the coal going to gasification the different coal sources are blended via a stacking procedure. Two different levels of blending takes place. First, each heap in the stack yard consists of more than one mine, and second the mines are stacked on the heap in several layers. Figure 2.12 shows a photograph of the heaps on the stack yard.

The type of stacking done in Secunda is chevron-strata stacking, where the coal from the different mines is longitudinally stacked in layers on top of each other (Mabuza, 2011). For example, if the target length of the heap is 120 meters, the stacker will throw coal from position 0 to position 120 and then

Figure 2.12: Stack yard



(a) *Heap being stacked in a layered fashion with coal from mines A and C*

(b) *Fully stacked heap with coal from mines A, B, C and E*

(c) *Blending performed by reclaimer slicing heap*

(d) *Heap being reclaimed further*

Figure 2.13: Stacking and reclaiming

Figure 2.14: Stacker

immediately from 120 to 0. Figure 2.13 (taken from Conradie (2007)) gives a high level picture of stacking and reclaiming. Figure 2.13(a) demonstrates the example where the stacker throws mine A on the heap first, and then after approximately two and a half layers starts throwing mine C on the heap. Figure 2.13(b) illustrates a finished heap containing four mines (A, B, C, E). This diagram is a simplification, as in reality several more layers of coal are deposited on a heap. Figure 2.14 shows a photograph of a stacker at the SCS stack yard which is in the process of throwing a layer of coal on the heap (stacking). An important detail of the stacking philosophy is that the stacker is not stopped when the conveyor is empty. It will therefore keep on moving forwards and backwards across the heap at a constant speed. The next layer of coal will commence wherever the stacker is located when it arrives on the feeding conveyor. Each stack yard contains three stackers.

In contrast with stacking, reclaiming starts from one end of the heap, and reclaims vertical segments of the heap as illustrated in Figures 2.13(c) and (d). The reclaimers are track-bound electrical powered machines (Mabuza, 2011). Although the reclaimers can move in both directions, they can only reclaim a heap from one direction until the heap is finished. There are three reclaimers at each stack yard. Figure 2.15 depicts a reclaimer busy reclaiming coal from a heap. Each reclaimer can reclaim approximately 2000 tons of coal per hour. The factory demand is however in the region of 2500 tons an hour, and therefore two heaps (at least) are reclaimed simultaneously. Normally there will be a primary heap reclaimed at 1500 tons an hour, and a secondary heap reclaimed at 1000 tons an hour. This brings about additional homogenization possibilities. If the properties of the reclaimed coal from the heaps can be

Figure 2.15: Reclaimer

predicted over the length of the heaps, the operator can ensure that a heap with non optimal coal qualities is reclaimed at lower volumes. In addition, a good coal quality heap can be reclaimed simultaneously to mitigate the effect of the lower quality coal on gasification.

The combination of stacking horizontally and reclaiming vertically leads to homogenization of the coal. The actual combination of mines and the number of layers and sequence of the mines will however impact the homogeneity of the resulting reclaimed coal. The scheduling of the mine blends has been discussed previously in Swart (2005) and Conradie (2007).

Given that the stacker moves at a constant speed there are three levers to manipulate the layers of the mines (blending) on the heap:

1. The length of the heap.
2. The tons/meter of coal on the conveyor.
3. The sequence of feeding the mines.

The length of the heap will impact the layers in that a longer heap will have fewer layers of any specific mine for each segment of the heap. A shorter heap will have the opposite effect. Demand constraints, as well as equipment constraints do however limit the length of the heaps that can be build at any given time.

Theoretically, depositing fewer tons per hour on the conveyors from each mine will have the effect of thinner layers and therefore more layers of each mine on the heaps. There are however demand constraints on the tons per hour.

The sequence of stacking the mines may be used as an additional lever to modify the heap profile. Due to variability within the mines it could be beneficial to have more layers of mines inter dispersed on the heaps. One cause of variability in coal qualities within mines is that each mine consists of different coal seams. The coal properties of each seam can differ significantly.

2.3.2 XRF and Material Movement File Data Workup

The information in the material movement file (for example Figure 2.6 for heap 14755) is used in combination with ash data from the XRF to create a profile of ash values over time for each mine. Additional elements are calibrated on the XRF, but as the calculations are similar only ash measurements will be discussed in this section. The information provides an indication of the ash composition of the heap with regards to the mines, which is used in the stacker simulation model discussed in Section 2.3.3. An example of the composition of the mines in the blends on Stacker 4 over a 28 day period is provided in the barplot in Figure 2.16. It is clear that over this period the blends on Stacker 4 were largely composed of Mine A and Mine B coal. In contrast a similar histogram for Stacker 1 (Figure 2.17) reveals that the blends were predominantly composed of Mine D, Mine C and Mine B coal.

The mapped data are stored in the database in a table similar to Figure 2.18 for efficiency.

2.3.3 Stacker Simulation Model

Various modeling methodologies have been proposed in literature for the modeling of stack blending problems. These include Discrete Element Models, CFD Models, and various other physics related methodologies (Cipold, 2013). These models although very accurate are unfortunately not computationally fast enough to be used in a real time monitoring environment. It was therefore deemed necessary to develop a stacker simulation model that was fast enough to be used in real time, but also of high accuracy.

A first step to modeling the stacker is to prepare the input data i.e., data on the run of mine coal from the different conveyors feeding into the stacker conveyor. From Figure 2.6 it is clear that the resolution of the data is not appropriate for a stack simulation model. Coal on the conveyor is a continuous process with variation. It is possible to approximate a continuous process by breaking it up into small discrete time buckets. Choosing the appropriate size buckets for the simulation is a trade off between efficiency and accuracy. Smaller time buckets increase the accuracy of the model, but also increase simulation time as well as memory use.

For the current simulation, buckets of one second were chosen for various reasons. First, the maximum amount of coal passing on the conveyor is 2000 tons/hour. This translates to approximately 500kg per second, which is a

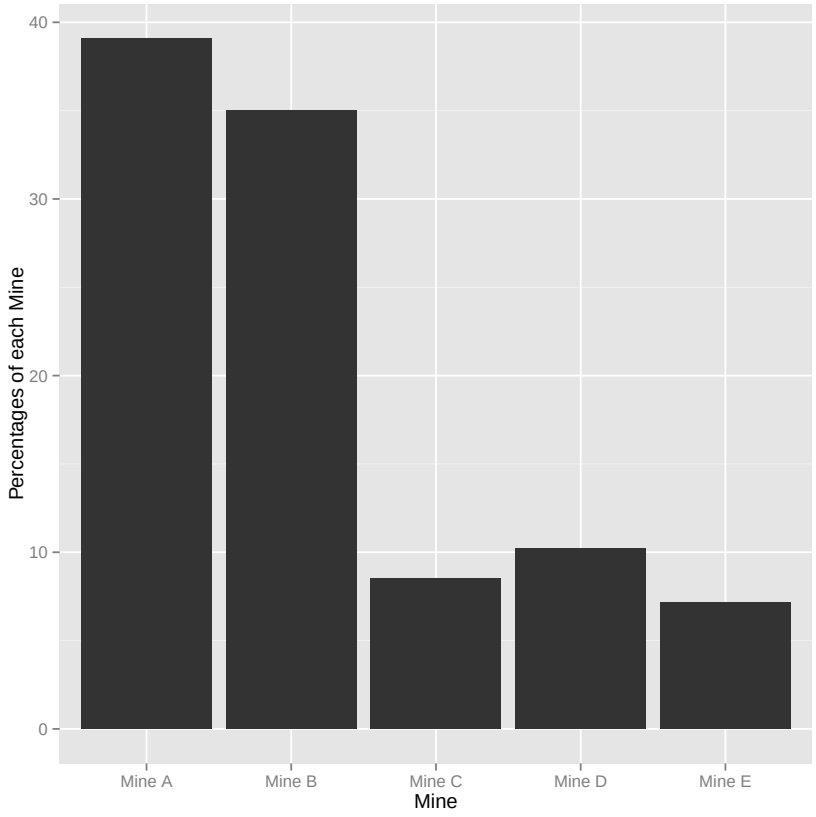


Figure 2.16: Histogram of mine percentages to Stacker 4

substantial amount of coal. Second the stacker speed is 7 s/m and a heap can be as short as 80 meters. A bigger time bucket will therefore lead to blocks of more than a meter in the simulation, and less than 100 actual points in the resulting stack. An additional benefit from a complexity perspective is that the base unit is seconds for the time format. After some experimentation it was concluded that the chosen time is acceptable.

A second challenge in using the material movement files for the simulation was the issue of variability. If the data are used as is, and just broken up into small time buckets, the tonnages will be constant for long periods, which will lead to a lot less variability in the simulated reclaimed coal than the actual reclaimed coal. This will not reflect reality. The actual distribution of the coal tonnages is unknown. However, after some preliminary investigation of the conveyors it was determined that the coal is distributed uniformly over the belt due to the natural packing of the coal transported over long distances on the conveyors from the mines. It was therefore decided to use the uniform distribution to simulate the coal tonnages. To conform to the actual mass balance the tonnages were simulated as a composition i.e., the total of the buckets for a time period is equal to the tonnages in the material movement file for the time period.

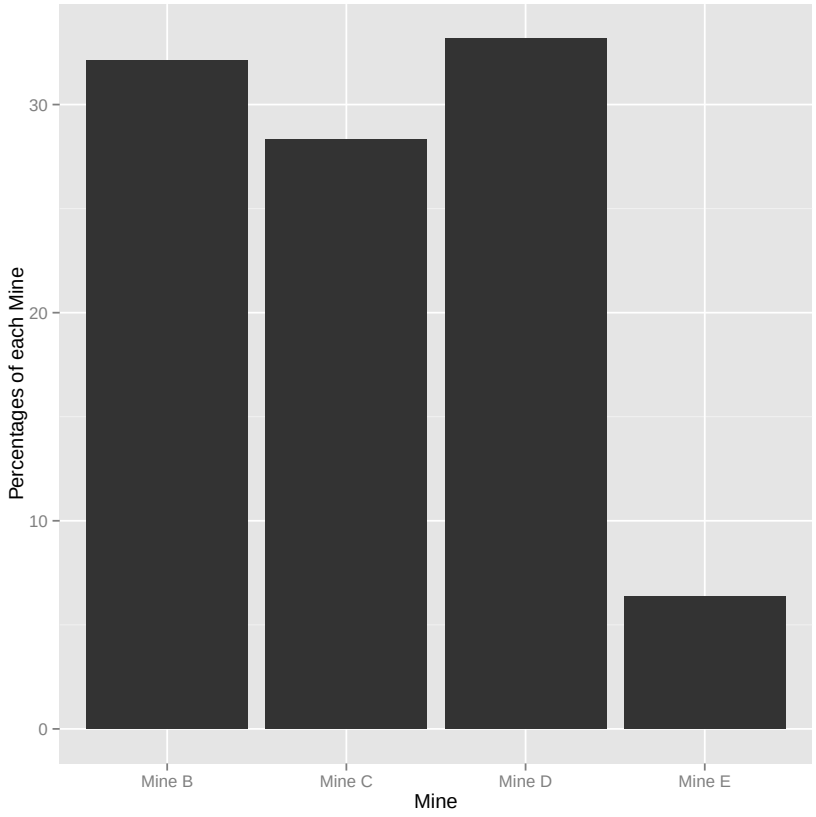


Figure 2.17: Histogram of mine percentages to Stacker 1

The algorithm to generate random compositional data is as follows:

Algorithm 2.1

- Draw random $u_i \sim \text{Uniform}(0,1), \quad i = 1, \ldots, n-1$.
- Sort u_i from smallest to largest i.e.,

$$\boldsymbol{u}^T = (u_{(1)}, u_{(2)}, \ldots, u_{(n-1)}).$$

- Specify $\boldsymbol{v}^T = (0, \boldsymbol{u}^T, 1)$, thus $\boldsymbol{v}$ is of dimension $(n+1) \times 1$.
- Let $d_i = v_{i+1} - v_i, \quad i = 1, \ldots, n$. Thus $d_i \sim \text{Uniform}(0,1)$ and $\sum_i d_i = 1$.
- Therefore, any quantity x can be represented by n uniformly distributed variables d_i through the identity

$$\sum_i d_i x = x \tag{2.3.1}$$

The R code used is:

```
##' Create a uniform composition
##'
##' @param n : Number of buckets
##' @param x : Total to normalize composition to
##'
##' @export

unifbrackets <- function(n,x=1){
    v <- c(0,sort(runif(n-1)),1)
    d <- v[2:(n+1)]-v[1:n]
    return(d*x)
}
```

To illustrate the simulation process one heap (Heap 15374) will be considered. It will be illustrated how the heap is tracked from the material movement files to a completed heap on the stockpile yard. In the following discussion one replication will be considered. However, since the process involves random distributions at least 10 replications are performed in the application of the methodology.

Consider the material movement file in Table 2.1 for Heap 15374. Each row contains the tons of coal (Weight) from a specific mine that passed over the conveyor between Start Date and End Date. The time stamps are in the POSIX time format denoting number of seconds from 1 January 1970. Note that although the material movement file captures the coal movement on the conveyor, the coal is stacked on a specific heap, and the information is therefore directly applicable to the heap as well.

Let x_{ij} be the weight in tons for row $i = 1, \ldots, I$ and codified mine $j = 1, \ldots, J$ (Table 2.3). Note, $I = 31$ for Heap 15374 in Table 2.1. Also, $J = 5$ for the mines transferred on the conveyor for Heap 15374. In addition denote the Start Time for row i by t_{ib} and the End Time by t_{ie}. Therefore, for each i, x_{ij} is the tons of mine j passed on the conveyor to Heap 15374 from time t_{ib} to t_{ie}. Let $k_{ij} = t_{ie} - t_{ib} + 1$ i.e., the number of seconds for row i that x_{ij} tons of mine j is transported on the conveyor to Heap 15374.

- Applying Algorithm 2.1, the tons of coal on the conveyor for each second z in interval k_{ij} can be simulated as $c_{zij} = d_{zij}x_{ij}, \quad z = 1, \ldots, k_{ij}$ and $d_{zij} \sim \text{Uniform}(0, 1)$ with $\sum_{z=1}^{k_{ij}} d_{zij}x_{ij} = \sum_{z=1}^{k_{ij}} c_{zij} = x_{ij}$.
- Note that $\boldsymbol{c}_{ij}$ is therefore a $k_{ij} \times 1$ vector of simulated coal tons for row i and mine j for each one second interval on the conveyor.
- Let $T_b = t_{1b}$ and $T_e = t_{Ie}$ and $M = T_e - T_b + 1$.
- A $M \times 1$ vector $\boldsymbol{m}$ consisting of all the one second buckets on the conveyor going to Heap 15347 can then be defined.

DateTime	SpectraID	Value	Mine
1401273216	480496	38.6053009033203	Mine D
1401273312	480497	38.5434989929199	Mine D
1401273409	480498	38.4618988037109	Mine D
1401273505	480499	38.7731018066406	Mine D
1401273602	480500	38.8815002441406	Mine D
1401273698	480501	39.464599609375	Mine D
1401273795	480502	38.7546005249023	Mine D
1401273891	480503	39.61669921875	Mine D
1401273988	480504	39.8130989074707	Mine D
1401274084	480505	39.4426002502441	Mine D
1401274179	480506	38.1879997253418	Mine D
1401274272	480507	36.9067001342773	Mine D
1401274367	480508	34.3428993225098	Mine D
1401274463	480509	32.6224994659424	Mine D
1401274655	480511	31.3579998016357	Mine C
1401274753	480512	31.136999130249	Mine C
1401274849	480513	31.2227001190186	Mine C
1401274946	480514	31.5263004302979	Mine C
1401275043	480515	31.3360004425049	Mine C
1401275140	480516	31.1546993255615	Mine C
1401275236	480517	30.8323993682861	Mine C
1401275330	480518	30.9032001495361	Mine C
1401275426	480519	30.4536991119385	Mine C

Figure 2.18: Table of mapped ash and mine data

Table 2.1: Material Movement File for Heap 15374

Row	SCS_ID (1)	Mine (2)	Start Time (3)	End Time (4)	Weight (ton) (5)
1	15374	Mine D	1409242890	1409242950	68.87
2	15374	Mine D	1409243072	1409243550	254.14
3	15374	Mine D	1409243640	1409245441	970.00
4	15374	Mine D	1409245440	1409245441	105.73
5	15374	Mine D	1409245470	1409248051	1300.77
6	15374	Mine A	1409248710	1409260410	5876.08
7	15374	Mine A	1409261220	1409263144	962.36
8	15374	Mine A	1409263140	1409263144	75.46
9	15374	Mine A	1409263175	1409271032	3940.58
10	15374	Mine C	1409271540	1409272170	288.12
11	15374	Mine C	1409272470	1409272623	42.14
12	15374	Mine C	1409274360	1409281470	3356.50
13	15374	Mine B	1409288911	1409290953	979.54
14	15374	Mine B	1409291250	1409300460	4438.46
15	15374	Mine B	1409303700	1409304720	473.86
16	15374	Mine B	1409304720	1409304720	89.44
17	15374	Mine B	1409304750	1409306190	508.26
18	15374	Mine B	1409306160	1409306190	53.32
19	15374	Mine B	1409306220	1409306911	18.92
20	15374	Mine E	1409308830	1409310420	244.02
21	15374	Mine B	1409310990	1409315164	1960.80
22	15374	Mine B	1409315610	1409315670	2.58
23	15374	Mine E	1409316120	1409316870	120.54
24	15374	Mine E	1409317653	1409321731	144.06
25	15374	Mine B	1409318520	1409318640	30.10
26	15374	Mine B	1409318732	1409321731	1402.66
27	15374	Mine B	1409324790	1409325111	125.56
28	15374	Mine B	1409453460	1409458590	2453.58
29	15374	Mine B	1409458981	1409459521	239.08
30	15374	Mine E	1409510580	1409516156	1382.78
31	15374	Mine B	1409515950	1409516156	67.94

- Each $\boldsymbol{c}_{ij}$ will occupy $\boldsymbol{m}_{(t_{ib}-T_b+1)}$ to $\boldsymbol{m}_{(t_{ie}-T_b+1)}$.
- Note that the $\boldsymbol{m}_{(t_{ie}-T_b+2)}$ to $\boldsymbol{m}_{(t_{(i+1)b}-T_b)}$ values for $i = 1, \ldots, I-1$ will be occupied by zero values.

For example, for Heap 15437, $T_b = 1409242890$ and $T_e = 1409516156$. Therefore, $M = T_e - T_b + 1 = 273267$. For $i = 1$, $(t_{1b} - T_b + 1) = 1409242890 - 1409242890 + 1 = 1$, and $(t_{1e} - T_b + 1) = 1409242950 - 1409242890 + 1 = 61$, and $m_{(1)}$ to $m_{(61)}$ will therefore populate $\boldsymbol{c}_{1j}$. Similarly, for $i = 1, (t_{ie} - T_b +$

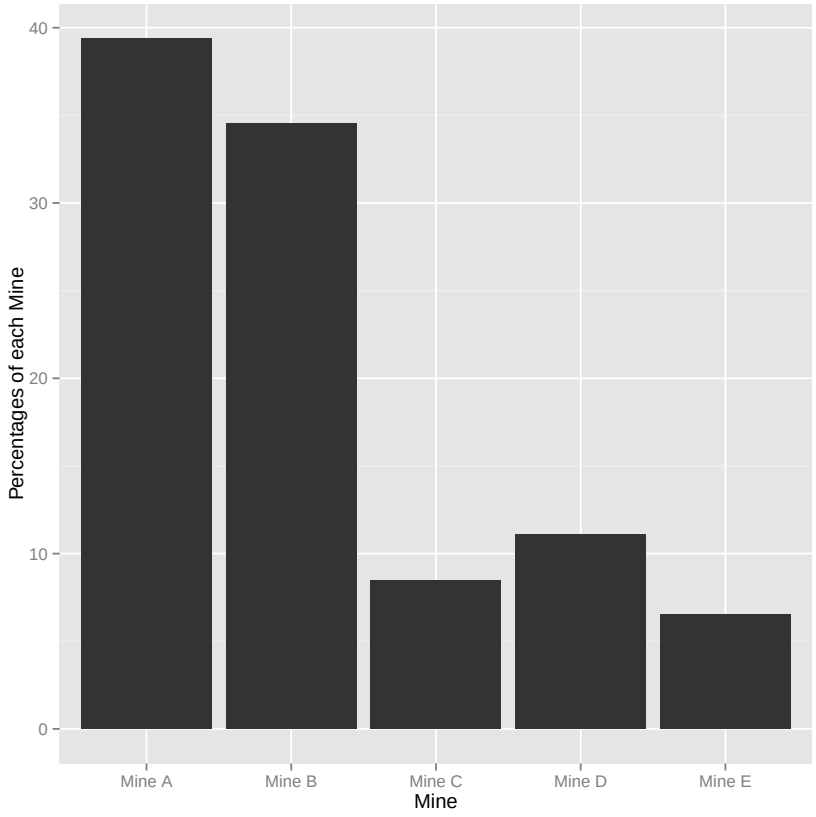


Figure 2.19: Histogram of mine percentages on Heap 15374

Table 2.2: CSRP information for Heap 15374

SCS_ID	15374
Length	160
Planned Tons	32000
Stacker	4
Start Position	140
End Position	300
Reclaim Direction	1

2) = 1409242950 − 1409242890 + 2 = 62 and $(t_{(1+1)b} - T_b)$ = 1409243072 − 1409242890 = 182, therefore $\boldsymbol{m}_{(62)}$ to $\boldsymbol{m}_{(182)} = 0$.

The total tons of coal can be calculated by $\sum_i x_{ij}$ and for Heap 15374 $\sum_i x_{ij} = 31976$ compared to 32000 planned from Table 2.2. The total tonnages for each mine j can also be calculated as follows:

$$T_j = \sum_i \begin{cases} x_{ij}, & \text{if } j_i = j \\ 0, & \text{otherwise} \end{cases}$$

Table 2.3: Mine numbers for Heap 15374

Number	Mine
1	Mine D
2	Mine A
3	Mine C
4	Mine B
5	Mine E

and the mine percentages as

$$P_j = \frac{T_j}{\sum_j T_j} \times 100 \tag{2.3.2}$$

The histogram of calculated mine percentages are shown in Figure 2.19 for Heap 15347.

The next step is to calculate the coal property values. Only ash content will be discussed, as the calculation of the other properties will be similar. As discussed in Section 2.3 the current XRF analyzer is situated on the conveyor to Stacker 4 on the Eastern factory. The Eastern factory receives coal from all six collieries. This information can be used to build a real time ash profile for all six collieries.

The XRF data are captured on a time stamp and an ash value is recorded every 90 seconds when coal is traveling on the conveyor to Stacker 4. Using the material movement data, the mine data can be added to the XRF data by merging the data on the time stamps. This will result in a table with a time stamp, an ash value, and a mine ID in each row (See Figure 2.18). As discussed in Section 2.3 this table is stored in the database for efficiency.

For utilization in the stacking simulation, the XRF data must be relevant to the specific heap, and an appropriate resolution. The first issue is mostly relevant for the stackers where no XRF analyzer is installed. To some extent it is also relevant to Stacker 4, as some gaps are present in the data due to calibration issues and drifting of the baseline. In addition, the 90 seconds resolution of the XRF data, although high resolution, is not appropriate for the stacking model if inferences need to be made about the distribution of ash over the heap length. As discussed above the stacker moves at 7 s/m, and 90 seconds will therefore represent a constant value for 12.86 meters. A decision was therefore made to use the distribution of the ash values of the collieries to simulate the distribution of the ash values of the coal on the conveyor.

The first step to generate the distribution is to find the appropriate range of data from the stored XRF data. The data must be as close as possible in time to the dates in the material movement file. It is however possible that coal from a specific mine has not been scheduled to go to the Eastern side for a number of days, and the appropriate data could be out of sequence with the times in the material movement files for a week or longer. A default of two

working days from the current start time in the material movement file are used to start the search. A check is performed if any data is available for the specific mine, and whether enough data points are available. If not, the start date is moved back a further 24 hours into history, and the check is performed again. This is repeated until the appropriate number of data points (currently 30) is obtained.

To sample data from the available data, it is assumed that the data are normally distributed. Considering the histograms of ash distributions depicted in Figure 2.21 for the run of mine (ROM) coal, the assumption of normality seems reasonable. Therefore, we sampled the appropriate number of points from the normal distribution to complement the one second material movement data. A histogram of the ash data for the mines for a 90 day period is provided in Figure 2.21a, and the histograms for the individual mines for the same period are depicted in Figures 2.21b to 2.21f. It is clear from Figure 2.21a that the XRF analyzer is capable of distinguishing between the individual mines. Note that for Mine D (Figure 2.21e) and Mine E (Figure 2.21f) the ash values above 45% are filtered on the analyser. It can be observed that the distribution of ash per mine is approximately normally distributed.

The normal distribution is described by two parameters; the population mean μ and the population variance σ^2 ($\mathcal{N}(\mu, \sigma^2)$). The population mean and variance for each mine j can be estimated by the sample mean ($\bar{x}_j$) and variance s_j^2 from the XRF data as shown in Table 2.4. The goal is to simulate k_{ij} ash values for each row i in Table 2.1. Let v_{zij} be a sample of k_{ij} values from $\mathcal{N}(\bar{x}_j, s_j^2)$. Therefore, $\boldsymbol{v}_{ij}$ will be a $k_{ij} \times 1$ vector of simulated ash values.

A $M \times 1$ vector $\boldsymbol{a}$ consisting of all the one second buckets of ash tons on the conveyor going to Heap 15347 is defined by multiplying elementwise the ash percentages in $\boldsymbol{v}_{ij}$ with the coal tonnages in $\boldsymbol{c}_{ij}$. Each $\boldsymbol{v}_{ij} \cdot \boldsymbol{c}_{ij}$ will occupy $\boldsymbol{a}_{(t_{ib}-T_b+1)}$ to $\boldsymbol{a}_{(t_{ie}-T_b+1)}$. Note that the $\boldsymbol{a}_{(t_{ie}-T_b+2)}$ to $\boldsymbol{a}_{(t_{(i+1)b}-T_b)}$ entries are occupied by zero values for $i = 1, \ldots, I-1$.

Let $\boldsymbol{m}$ be the vector of coal tonnages, and $\boldsymbol{a}$ the vector of ash tonnages passing on the conveyor in one second. As discussed previously the stacker moves at a constant speed of $r = 7s/m$. If we define the length of the heap as L (From Table 2.2 $L = 160m$ for heap 15374) meters, and the length of $\boldsymbol{m}$ as M ($M = 273267$ for heap 15374), the number of layers R on the heap can be specified as

$$R = \lceil M/(L \times r) \rceil \tag{2.3.3}$$

where $\lceil x \rceil$ is defined as ceiling(x) (the smallest integer not less than x). Let $l = L \times r$, then the simulated heap can be defined as the $R \times l$ matrix $\boldsymbol{H}$. This is an obvious oversimplification as coal particles will not stack perfectly, and in practice the ends of the heap will not be square, but for the sake of simplicity the assumption of a rectangular heap will suffice. As the stacker moves forward and backwards building the heap, the stacking can be envisioned as a folding of the coal on the conveyor into multiple layers. The length of each fold will

be l, and the number of folds will be R. As R was rounded up it is necessary to add $R \times l - M$ empty one second buckets to the front of vectors $\boldsymbol{m}$ and $\boldsymbol{a}$. Each row r of $\boldsymbol{H}$ will be populated as follows:

$$\boldsymbol{H}_{r(1,\ldots,l)} = \begin{cases} \boldsymbol{m}_{(r\times l)+1,\ldots,(r\times l)+l}, & \text{if } r \text{ is odd} \\ \boldsymbol{m}_{(r\times l)+l,\ldots,(r\times l)+1}, & \text{if } r \text{ is even} \end{cases}$$

A $R \times l$ matrix of ash tonnages $\boldsymbol{A}$ is defined utilising $\boldsymbol{a}$ similar to $\boldsymbol{H}$. The ash percentage over the heap length can then be calculated by first calculating the tons ash ($\boldsymbol{ta}$) and ton coal ($\boldsymbol{th}$) for each row in $\boldsymbol{A}$ and $\boldsymbol{H}$.

$$\boldsymbol{ta}_i = \sum_{j=1}^{l} \boldsymbol{A}_{(ij)} \quad , i = 1, \ldots, R \tag{2.3.4}$$

$$\boldsymbol{th}_i = \sum_{j=1}^{l} \boldsymbol{H}_{(ij)} \quad , i = 1, \ldots, R \tag{2.3.5}$$

The percentage of ash over the length of the heap ($\boldsymbol{ap}$) can therefore be calculated as

$$\boldsymbol{ap}_i = \frac{\boldsymbol{ta}_i}{\boldsymbol{th}_i} \quad , i = 1, \ldots, R \tag{2.3.6}$$

The resulting output of the simulation for Heap 15374 is shown in Figure 2.20.

Table 2.4 contains the XRF information for Heap 15374. The overall average ash and standard error of the average ash of a heap can be calculated as follows:

- Let P_j be the proportion of mine j on the heap, w_j the tons of coal of mine j, $\bar{x}_j$ the average of the ash from the XRF analyzer for mine j, s_j the standard deviation of the ash from the XRF analyzer for mine j, and n_j the number of XRF measurements for each mine j.
- The overall average ash can be calculated as a proportional mean as follows:

$$\bar{X} = \sum_{j} \bar{x}_j \times P_j \tag{2.3.7}$$

- The standard error of the mean ash content for the heap can be calculated as follows:

$$\mathcal{SE} = \sqrt{\sum_{j} P_j^2 \times \frac{s_j^2}{n_j}} \tag{2.3.8}$$

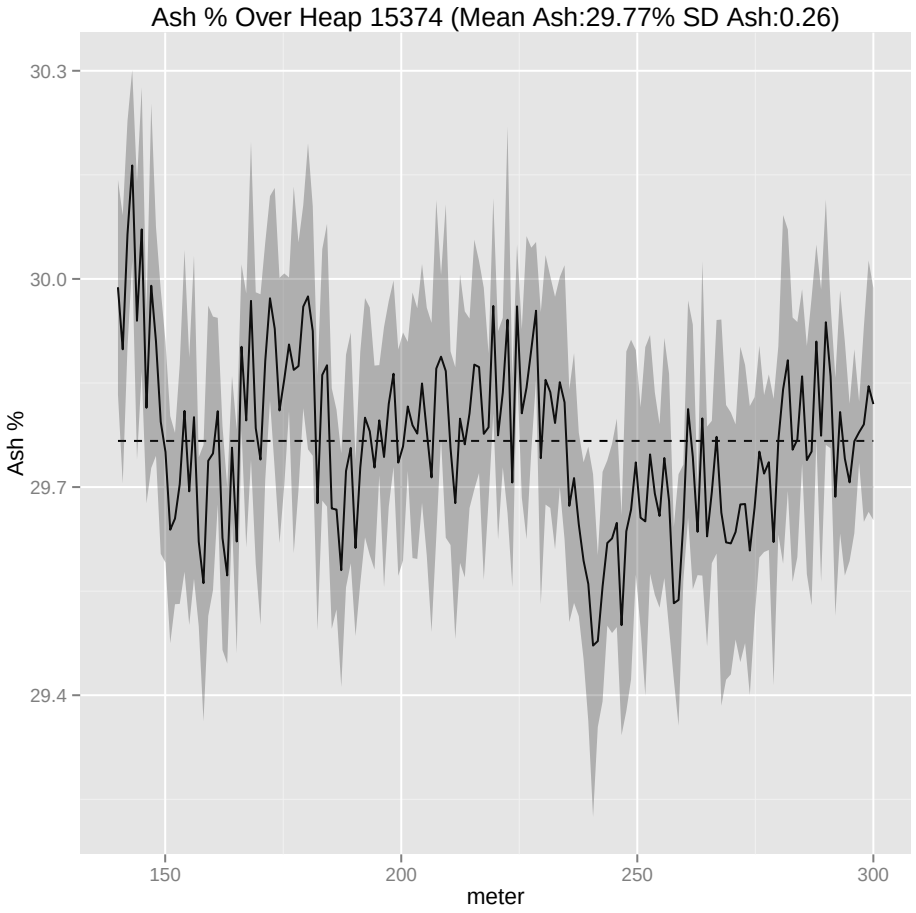
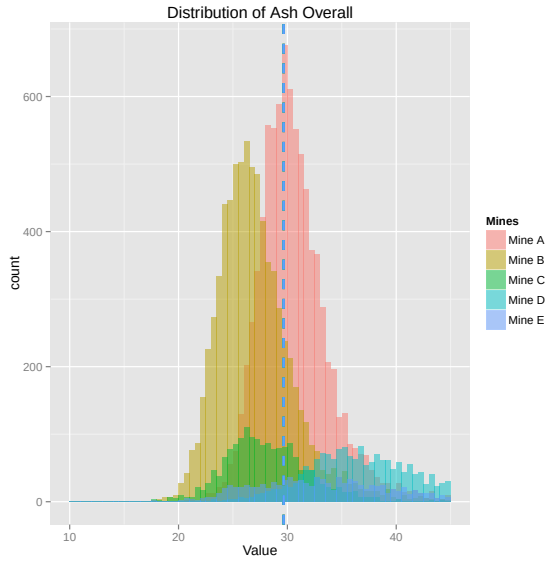


Figure 2.20: Ash Percentage over length for Heap 15374

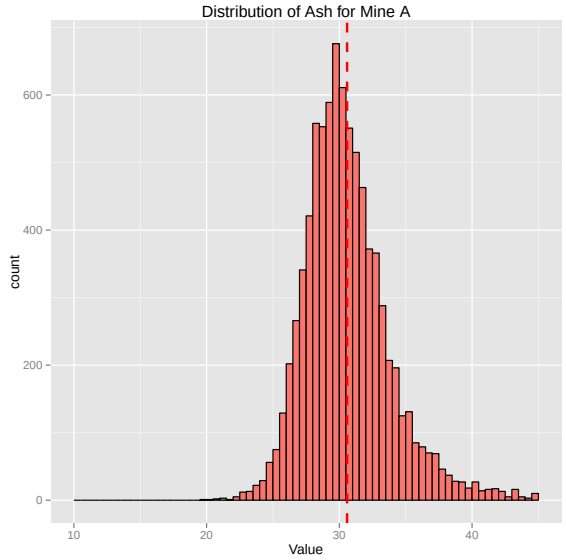
- The standard deviation of the ash for the heap can be calculated as follows:

$$\mathcal{SD} = \sqrt{\frac{\sum_j w_j^2 \times s_j^2}{(\sum_j w_j)^2}} \tag{2.3.9}$$

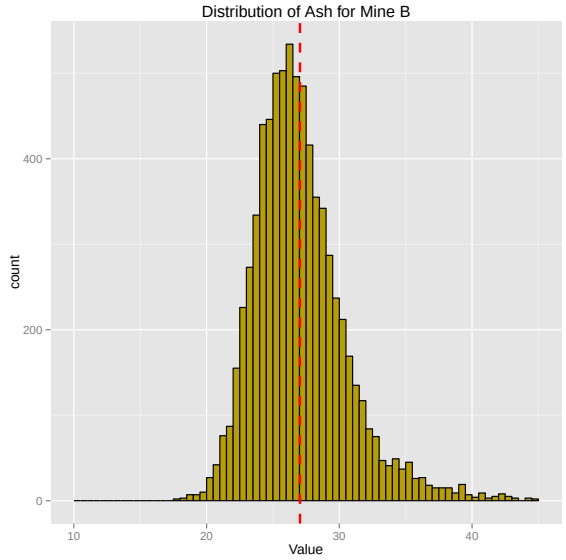
The results for Heap 15347 are shown in Table 2.5. Included are the planned and actual tons, the start date and end date for the heap as well as the length of the heap and tons per meter. Some additional information that can be obtained from the material movement information are the layering of the mines on the heap. Refer to Figure 2.22. The horizontal axis depicts the meters as defined in Section 2.1.4. The vertical axis depicts the layers over time for the heap. The mines are depicted by the colours in the legend. Note that the white layers depict the times when the conveyor were empty and the stacker was still moving. It is important to note that the layers depict the movement of the stacker over time and not the actual positions of the coal on the heap, as it does not include the tons on the layers. This graph provides important



(a) Histogram of Overall Ash

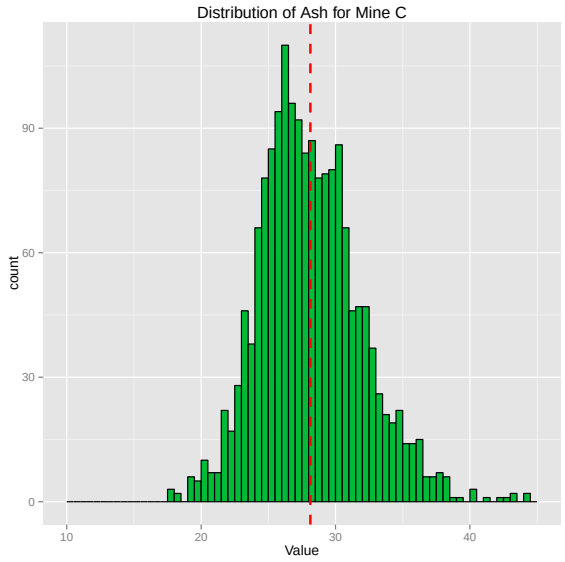


(b) Histogram of Mine A ash

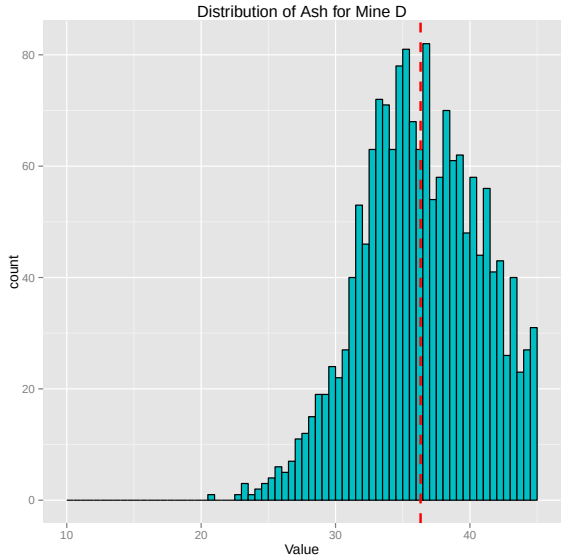


(c) Histogram of Mine B Ash

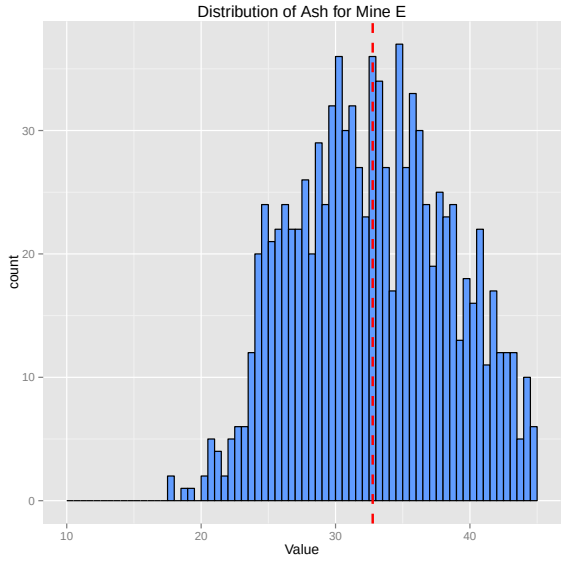
Figure 2.21: Overall as well as ROM ash distributions



(d) Histogram of Mine C Ash



(e) Histogram of Mine D ash



(f) Histogram of Mine E ash

Figure 2.21: Overall as well as ROM ash distributions continued

information to the business concerning the homogeneity of the coal properties on the heap.

Table 2.4: Mine properties for Heap 15374

Mines	Tons	Mean Ash	SD Ash	% in Blend	Data From	Data To
Mine A	10854	29.68	2.39	33.95	29-Aug-14 22:16	30-Aug-14 23:59
Mine D	2700	36.00	4.00	8.44	28-Aug-14 18:28	28-Aug-14 19:44
Mine C	3687	27.19	3.68	11.53	30-Aug-14 02:24	30-Aug-14 04:39
Mine E	1891	37.20	4.69	5.92	30-Aug-14 08:56	31-Aug-14 21:43
Mine B	12844	27.97	3.01	40.17	30-Aug-14 05:33	31-Aug-14 06:15

Table 2.5: Summary for Heap 15374

Heap Summary		
% Ash Summary	Average = 29.7%	Standard Error = 0.12%
Total Tons	Planned = 32000	Actual = 31976
Stack Date	Start = 28-Aug-14 18:21	End = 31-Aug-14 22:15
Total Heap Length	160 Meters	200 Tons/Meter

2.4 Reclaimer Simulation

As discussed in the previous section, the coal going to gasification is reclaimed from at least two heaps. Two different levels of prediction of reclaimed coal properties are currently implemented. The first level uses the average coal properties of the heaps to calculate the predicted coal quality to gasification. For this level of accuracy the following information is needed:

1. Real time information of the heap numbers being reclaimed.
2. Real time information on the tons reclaimed from each heap.
3. Average coal qualities of the reclaimed heaps.

The benefit of using the average heap properties is simplicity. However, it is possible that the quality of a heap can differ over the length of the heap. The reclaimed coal qualities may therefore be substantially different from the average in real time. Adding the actual position of the reclaimer to the prediction will increase the accuracy of the prediction, but at the cost of substantial additional complexity to the input required:

1. In the previous section the modeling philosophy was discussed for the stacker. Some assumptions were however implicit in the modeling approach:

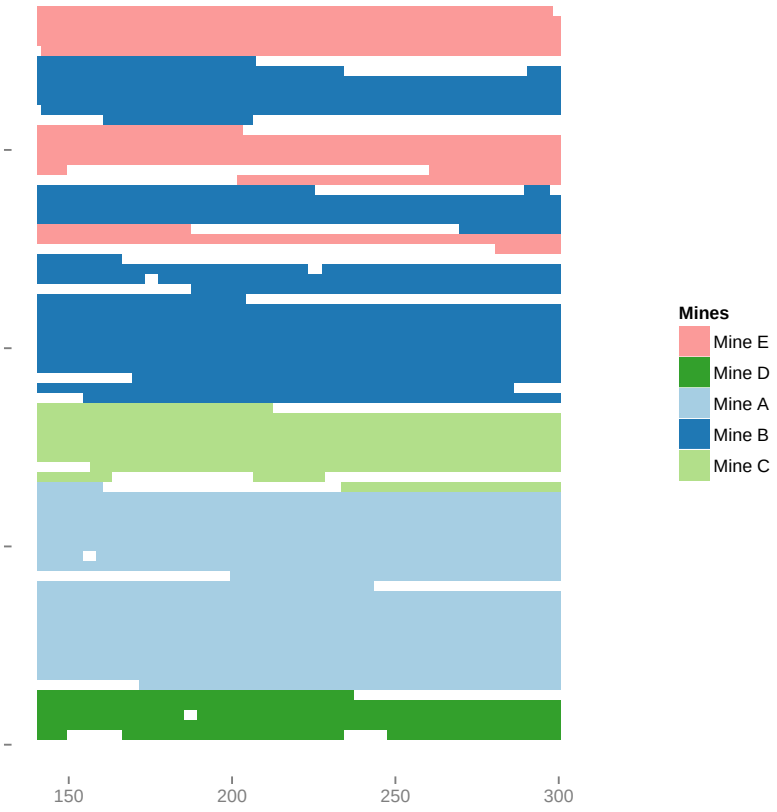


Figure 2.22: Mine layers over length for Heap 15374

a) The stacker will start stacking at the beginning of the heap.

b) The stacker speed is a constant 7 s/m.

c) The heap length information from the CSRP is accurate.

In practice, none of the above will be 100% accurate, and the position indicator is at best just an approximation. The only way to accurately predict the position of coal on the heap will be to add a GPS to the stacker (and reclaimer for the reclaiming). A project is currently underway to implement the GPS's on the stackers and reclaimers to assist in the reclaimer simulation, but this information was not available at the time of this study.

2. Knowledge about the side of the heap where reclaiming starts are required.

3. Knowledge about the actual position of the reclaimer in real time is required.

In the next section the real time implementation of predicting the average heap coal qualities will be discussed. The simulation of the reclaimed coal qualities to gasification over the length of the heap via user input will be discussed in Section 2.4.2

2.4.1 Real Time Reclaimed Coal Prediction

The complexity in the prediction of reclaimed coal qualities to gasification is largely a function of the availability of appropriate data. Currently two sets of data are available. The first set is similar to the stacking simulation data in the material movement files. An extract from the reclaiming material movement file is shown in Table 2.6. The reclaiming material movement file is part of the overall material movement file in the SCS database. The only difference is in one variable which indicates if the data are relevant to stacking or reclaiming as shown in Table 2.6. Each of the rows contain a SCS_ID, the start and end times, and the weight (tons) reclaimed. This information should be sufficient as it contains the first two of the three types of information required for the coal quality prediction (the third type of information is available from the stacker simulation from the previous section):

1. Real time information for the heap numbers reclaimed.
2. Real time information on the tons reclaimed from each heap.
3. Average coal qualities of the reclaimed heaps.

After a mass balance study was performed on this information it was however concluded that there are some serious problems in using this data for real time predictions. The initial reason for capturing the reclaiming data in the material movement file was to calculate the tons of coal available on the stock yard at discrete time points. In the material movement files corrections are frequently performed which are not logged against the actual time, but against the time of correction. This was deemed necessary because of inaccuracies on the scales measuring the reclaimed coal at the reclaimer.

The reclaimed coal tons are however measured at an alternative position on the conveyors, and captured on the PI DCS system. These scales are believed to be more accurate, and the engineers suggested the use of these measurements for the actual tonnages to gasification. Table 2.7 shows an extract from the reclaimed tonnages as captured on PI. These data were extracted and aggregated via the `ssldcsutils` package that will be discussed in detail in Chapter 5. Table 2.7 can therefore provide the tons of coal for each reclaimer, but the heap reclaimed from is not captured. The integration of the data in Tables 2.6 and 2.7 is therefore necessary to obtain the relevant information for the real time reclaimed coal prediction. Specifically, the heaps reclaimed for a specific time are extracted from Table 2.6, and combined with the tons coal

reclaimed from Table 2.7. In addition to the data in these two tables the data from the CSRP files are needed to assign a Stacker/Reclaimer to each heap.

Table 2.6: Material movement file for reclaiming

SCS_ID	Mine	Start Date	End Date	Weight
14487	SCS_COAL	01-Jan-14 01:07	01-Jan-14 13:09	13674
14483	SCS_COAL	01-Jan-14 03:18	01-Jan-14 03:21	92
14484	SCS_COAL	01-Jan-14 03:21	01-Jan-14 05:38	707
14485	SCS_COAL	01-Jan-14 06:24	01-Jan-14 06:39	242
14477	SCS_COAL	01-Jan-14 06:26	01-Jan-14 07:28	1642
14485	SCS_COAL	01-Jan-14 07:24	01-Jan-14 08:18	1025
14485	SCS_COAL	01-Jan-14 08:28	01-Jan-14 09:13	550
14485	SCS_COAL	01-Jan-14 09:23	01-Jan-14 09:59	636
14483	SCS_COAL	01-Jan-14 12:59	01-Jan-14 13:07	80
14476	SCS_COAL	01-Jan-14 13:07	01-Jan-14 17:08	5937
14484	SCS_COAL	01-Jan-14 13:51	01-Jan-14 13:52	40
14484	SCS_COAL	01-Jan-14 13:54	01-Jan-14 13:55	29
14485	SCS_COAL	01-Jan-14 14:00	01-Jan-14 16:00	1621
14484	SCS_COAL	01-Jan-14 14:08	01-Jan-14 14:12	68
14484	SCS_COAL	01-Jan-14 14:18	01-Jan-14 14:18	30

Table 2.7: Reclaimed tons from PI DCS system

DateTime	Recl 1	Recl 2	Recl 3	Recl 4	Recl 5	Recl 6
01-Jan-14 00:00	930	1754	75	716	1473	0
01-Jan-14 00:30	1132	974	423	624	1372	0
01-Jan-14 01:00	1455	78	1046	965	1386	0
01-Jan-14 01:30	1280	78	760	1392	1431	0
01-Jan-14 02:00	1224	79	778	1195	1395	0
01-Jan-14 02:30	1124	78	875	1096	1457	0
01-Jan-14 03:00	759	78	1241	1310	1075	0
01-Jan-14 03:30	624	1343	0	1156	1174	0
01-Jan-14 04:00	791	779	971	912	959	0
01-Jan-14 04:30	949	79	919	786	994	423

The `getdata` function from the `ssldcsutils` package retrieves the local copy of the DCS data and aggregates the data using various aggregation methods, and any time interval defined by the user. For the reclaimer prediction data an initial time interval of 30 minutes was chosen, but this can be changed easily if need be. A time weighted average aggregation (see Chapter 5 Section 5.5.1) was used.

Table 2.8: Material Movement combined with the CSRP data

SCS_ID	Start Date	End Date	Weight	Stack
14483	01-Jan-14 00:00	01-Jan-14 01:30	1107	3
14487	01-Jan-14 00:30	01-Jan-14 13:00	13674	5
14477	01-Jan-14 01:00	01-Jan-14 02:30	1099	2
14483	01-Jan-14 01:30	01-Jan-14 03:30	1400	3
14480	01-Jan-14 02:30	01-Jan-14 12:30	6955	6
14483	01-Jan-14 03:00	01-Jan-14 03:30	92	3
14484	01-Jan-14 03:00	01-Jan-14 05:30	707	4
14483	01-Jan-14 03:00	01-Jan-14 05:00	2124	3
14477	01-Jan-14 03:30	01-Jan-14 05:30	238	2
14481	01-Jan-14 04:00	01-Jan-14 05:00	517	1
14483	01-Jan-14 04:30	01-Jan-14 06:00	535	3
14485	01-Jan-14 04:30	01-Jan-14 06:00	1013	1
14484	01-Jan-14 05:00	01-Jan-14 06:00	8	4
14485	01-Jan-14 05:30	01-Jan-14 06:30	417	1
14483	01-Jan-14 05:30	01-Jan-14 06:30	557	3

The first step to get the material movement in an appropriate format to merge with the DCS data is to merge it with the CSRP derived data (Figure 2.9) to link the SCS_ID to a stacker/reclaimer, giving Table 2.8. In addition, the Start Date column was rounded down, and the End Date rounded up to the nearest 30 minutes. To merge Table 2.8 with Table 2.7 the data should be reshaped to have the stacks as columns, and the SCS_ID's in the stack 1 to 6 columns. The times must also be comparable to Table 2.7. An example of the desired output is shown in Table 2.9. To illustrate the reshaping algorithm, the R code is provided below where `mtmvgrp` refers to the data in Table 2.8.

```
n <- length(seq(min(mtmvgrp$startdate),max(mtmvgrp$enddate),by=30*60))

mtmvresh <- matrix(0,n,7)
colnames(mtmvresh) <- c("DateTime",paste("Stack",1:6,sep=" "))
mtmvresh[,1] <- seq(min(mtmvgrp$startdate),max(mtmvgrp$enddate),by=30*60)
for(i in 1:nrow(mtmvgrp)) {
    mtmvresh[(mtmvok[,1]>=mtmvgrp$startdate[i]) &
        (mtmvok[,1] <= mtmvgrp$enddate[i]),
           mtmvgrp$stack[i]+1] <- mtmvgrp$scsid[i]

}
```

Two transformations are performed. The first transformation is with regards to the start and end date columns. As there are more than one stack being reclaimed at the same time the start and end dates of the different rows

overlap. The data are transformed by creating a single regular time step (30 minutes), DateTime column, and using the start and end times to assign the SCS_ID to the appropriate Stack column using the Stack column in Table 2.8. The time step in the algorithm increments by 30×60 as the data are stored in seconds (refer to Chapter 5 Section 5.2). Table 2.9 is now in an appropriate format to merge with Table 2.7.

Table 2.9: Reshaped Material Movement and CSRP data

DateTime	Stack 1	Stack 2	Stack 3	Stack 4	Stack 5	Stack 6
01-Jan-14 00:00	0	0	14483	0	0	0
01-Jan-14 00:30	0	0	14483	0	14487	0
01-Jan-14 01:00	0	14477	14483	0	14487	0
01-Jan-14 01:30	0	14477	14483	0	14487	0
01-Jan-14 02:00	0	14477	14483	0	14487	0
01-Jan-14 02:30	0	14477	14483	0	14487	14480
01-Jan-14 03:00	0	0	14483	14484	14487	14480
01-Jan-14 03:30	0	14477	14483	14484	14487	14480
01-Jan-14 04:00	14481	14477	14483	14484	14487	14480
01-Jan-14 04:30	14485	14477	14483	14484	14487	14480
01-Jan-14 05:00	14485	14477	14483	14484	14487	14480
01-Jan-14 05:30	14485	14477	14483	14484	14487	14480
01-Jan-14 06:00	14485	14477	14483	14484	14487	14480
01-Jan-14 06:30	14485	14477	14483	0	14487	14480
01-Jan-14 07:00	14485	14477	14483	0	14487	14480

Three steps are necessary to merge the data.

- First, the actual tables must be merged. The dplyr (Wickham and Francois, 2014) `inner_join` function was used to merge the two tables (see code example below).
- Second, due to a time mismatch between the material movement data and the PI data, a correction must be applied to the material movement data.
- And last, the data must be filtered for very low or negative reclaimed values, and for missing data.

To synchronize the times of the data sets, the closest previous value from the material movement file is located and used. Extracts of the merged data sets are shown in Tables 2.10 and 2.11 for the Western and Eastern side respectively. These tables contain the relevant data that can be used in combination with the coal quality data from the stack simulation model to predict the reclaimed coal qualities going to gasification. This is done using the following R code:

```
pimtmv <- pirecl %>%
    dplyr::inner_join(mtmvresh)

minval <- 50
for(i in 1:nrow(pimtmv)) {
  for(stck in 1:6) {
    if(pimtmv[[paste("recl",stck,sep="")]][i] > minval) {
      if(pimtmv[[paste("stack",stck,sep="")]][i] == 0) {
         pimtmv[[paste("stack",stck,sep="")]][i] <-
         pimtmv[[paste("stack",stck,sep="")]][max(which(pimtmv[[paste("stack",
                                                stck,sep="")]][1:i]>0))]
       }

    }
  }
}
```

Table 2.10: Merged PI and Material Movement File for Western factory

DateTime	recl1	recl2	recl3	stack1	stack2	stack3
01-Jan-14 04:00	791	779	971	14481	14477	14483
01-Jan-14 04:30	949	79	919	14485	14477	14483
01-Jan-14 05:00	920	78	951	14485	14477	14483
01-Jan-14 05:30	1334	79	1236	14485	14477	14483
01-Jan-14 06:00	860	329	955	14485	14477	14483
01-Jan-14 06:30	635	245	1456	14485	14477	14483
01-Jan-14 07:00	1570	0	887	14485	0	14483
01-Jan-14 07:30	1057	73	550	14485	14477	14483
01-Jan-14 08:00	750	1185	972	14485	14477	14483
01-Jan-14 08:30	849	1593	0	14485	14477	0
01-Jan-14 09:00	899	1516	0	14485	14477	0
01-Jan-14 09:30	1312	484	241	14485	14477	14483
01-Jan-14 10:00	1247	1390	0	14485	14477	0
01-Jan-14 10:30	1653	890	519	14485	14477	14483
01-Jan-14 11:00	817	1503	236	14485	14477	14483

2.4.2 Simulated Reclaimed Coal to Gasification

The information in Table 2.10 and 2.11 can now be combined with the information obtained in Section 2.3.3 to predict in real time the coal properties going to the gasifiers. Specifically the tonnages in merged PI and material movement

Table 2.11: Merged PI and Material Movement File for Eastern factory

DateTime	recl4	recl5	recl6	stack4	stack5	stack6
01-Jan-14 04:00	912	959	0	14484	14487	0
01-Jan-14 04:30	786	994	423	14484	14487	14480
01-Jan-14 05:00	53	1044	1304	14484	14487	14480
01-Jan-14 05:30	111	1274	505	14484	14487	14480
01-Jan-14 06:00	654	1385	487	14484	14487	14480
01-Jan-14 06:30	131	1461	501	14484	14487	14480
01-Jan-14 07:00	398	1258	506	14484	14487	14480
01-Jan-14 07:30	66	1373	1027	14484	14487	14480
01-Jan-14 08:00	0	1507	1427	0	14487	14480
01-Jan-14 08:30	0	1548	516	0	14487	14480
01-Jan-14 09:00	479	1529	516	14484	14487	14480
01-Jan-14 09:30	829	1210	494	14484	14487	14480
01-Jan-14 10:00	623	1064	496	14484	14487	14480
01-Jan-14 10:30	437	1473	482	14484	14487	14480
01-Jan-14 11:00	776	781	1154	14484	14487	14480

data can be combined with the average ash for the heaps, and the standard deviation of the heaps to calculate a pooled average ash going to gasification using (2.3.7) and a pooled standard deviation using (2.3.9). An example of the predicted ash percentage of the reclaimed coal is shown in Figure 2.23.

2.5 Empirical Slag Model

In addition to the ash content, the slagging propensity of the ash is another important coal property to be able to predict and to monitor for the gasification process. Currently a computer model exists which predicts the slag formation of the ash as a function of coal properties and reactor temperature using the FactSage software (Bale *et al.*, 2009). However, the current model poses three challenges for inclusion in the real-time MSPEM™ application:

1. FactSage is only available on Microsoft Windows. It is also attached to a license per user. In contrast, the MSPEM™ application is hosted on a Linux server.

2. No convenient methodology exists to interactively interface with the software.

3. The model does not solve instantaneously.

The model can however be utilised to obtain an empirical metamodel that can be used in lieu of the current FactSage model. Design and Analysis of Computer Experiments (DACE) is concerned with finding the optimal points

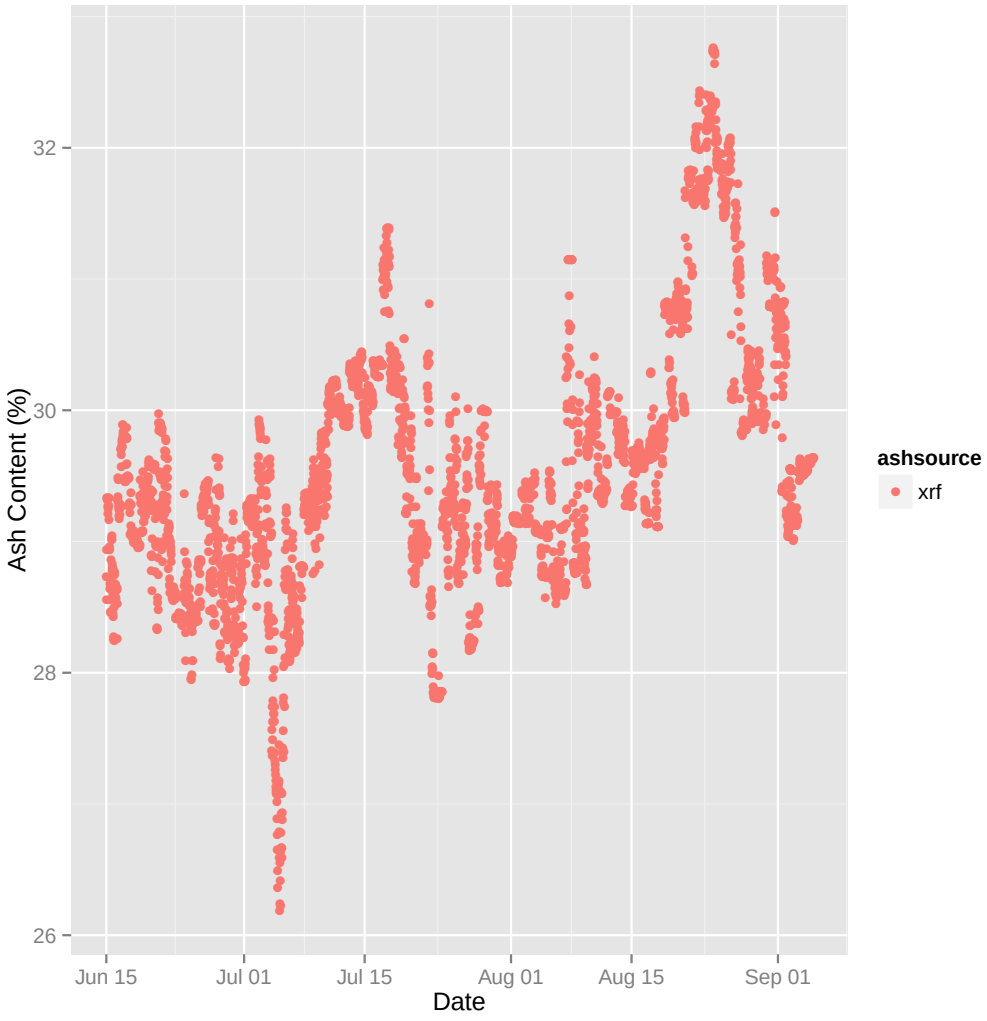


Figure 2.23: Predicted reclaimed coal ash percentage

in the design space to fit empirical models to highly complex computer models (Sacks *et al.*, 1989; Santner *et al.*, 2003). Generally, space filling designs are employed for computer experiments. However, the input to the slag model, specifically the ash composition, is a composition. In the next section space-filling designs for mixture experiments will be discussed and demonstrated for the current application. This work was published in Coetzer *et al.* (2012).

2.5.1 Efficient Maximin Distance Design for Experiments in Mixtures

Coetzer *et al.* (2012) stated that computer models are becoming an integral part of process design and decision making in industry. These models are generally non-linear with many input and output variables. In addition, the

model outputs are deterministic i.e., a given set of input parameters will always map to the same set of outputs. These properties necessitate designs that are applicable to a wide range of models. The recommended design strategy is to employ space-filling designs for running the computer code (Fang *et al.*, 2006; Santner *et al.*, 2003).

In the petrochemical industry the inputs to computer models are generally a mixture of chemical components or molecules $x_1, x_2, \ldots, x_s$ such that

$$\sum_{i=1}^{s} x_i = 1 \tag{2.5.1}$$

and each $x_i \in [0, 1]$. Furthermore, in industrial processes the mixture variables are subject to lower and upper constraints, i.e.

$$0 \leq L_i \leq x_i \leq U_i \leq 1, \qquad i = 1, 2, \ldots, s, \tag{2.5.2}$$

where L_i and U_i are the lower and upper limits respectively. These constraints commonly result in a highly constrained region where each component is present in a very narrow range (Borkowski and Piepel, 2009).

Several space-filling designs exist in literature (Fang *et al.*, 2006; Santner *et al.*, 2003). In the specific domain of mixture design and analysis of computer experiments (MDACE) these designs must be narrowed down to those that can handle arbitrarily constrained input parameters. Stinstra *et al.* (2003) noted that there are three space-filling criteria commonly found in literature; maximin, minimax, and integrated mean square error (IMSE). The paper then stated that the minimax and IMSE criteria are computationally very difficult for arbitrarily constrained regions, and that the maximin criterion is therefore proposed as the criterion of choice. However, Stinstra *et al.* (2003) did not consider mixture variables. Borkowski and Piepel (2009) considered space-filling designs for mixture variables. Specifically they generated uniform designs in the constrained simplex. However, they used only Euclidean distances to evaluate the different designs generated, and did not consider any other dissimilarity measures. Coetzer *et al.* (2012) considered the maximin criterion for constructing space-filing designs for mixture variables subject to constraints (2.5.1) and (2.5.2) with different dissimilarity measures.

Let $\mathcal{X}$ be the design space for s variables such that the constraints in (2.5.1) and (2.5.2) are satisfied. $D(n, s)$ is a maximin distance design with n experiments if, for $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{X}$:

$$\min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in D(n,s)} \rho_2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \max_{D \in \mathcal{X}} \min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in D} \rho_2(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2.5.3}$$

where $\rho_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is some dissimilarity criterion (Johnson *et al.*, 1990). In literature the Euclidean distance

$$[\sum_{k=1}^{s} (x_{ik} - x_{jk})^2]^{\frac{1}{2}} \tag{2.5.4}$$

is generally used as a measure of dissimilarity. However, there exists a rich body of literature regarding dissimilarity measures and their properties (see for example Cox and Cox (2001); Gower (1982); Gower and Legendre (1986)). The effect of different dissimilarity measures ρ_2 on the properties of the maximin design was not previously discussed in the literature.

Objective and robust design criteria should be employed for the evaluation of designs. Specifically, any criterion to evaluate the design should not be biased towards any of the dissimilarity measures investigated, and should itself be able to operate over the total design space. The criterion should also be able to discriminate between different designs, i.e. be sensitive to subtle changes in the designs.

Although computationally less difficult than minimax and IMSE designs, maximin designs remain very difficult optimization problems to solve (Stinstra *et al.*, 2003). Maximin designs possess various properties that either increase the complexity or improve the computational efficiency of the optimisation.

Dissimilarity measures are discussed in Section 2.5.1.1. In Section 2.5.1.2 an algorithm for obtaining a maximin design given a specific dissimilarity measure is presented. In Section 2.5.1.3 some criteria to evaluate and compare the different maximin designs are discussed.

2.5.1.1 Dissimilarity Measures

Most of the criteria for space-filling designs rely on some measure ρ_2 of dissimilarity between two design points either directly, as part of the design criterion as in, for example, the maximin and minimax criteria, or as in a design evaluation criterion (Borkowski and Piepel, 2009). In the design literature the Euclidean distance ($\rho_2 = [\sum_{k=1}^{s}(x_{ik} - x_{jk})^2]^{\frac{1}{2}}$) is almost exclusively used as the measure of dissimilarity.

Equations (2.5.6) - (2.5.12) depict different dissimilarity measures commonly encountered in the literature. In the Aitchison measure $g(\boldsymbol{x}_i)$ is defined as the geometric mean of the composition, i.e. $g(\boldsymbol{x}_i) = (\prod_{k=1}^{s} x_{ik})^{\frac{1}{s}}$, and is discussed extensively in Aitchison (1992). It is equivalent to the Euclidean distance performed on additive log ratio transformed (ALR) data (Aitchison, 1986). The ALR transformation is defined as:

$$z_i = log\left(\frac{x_i}{x_D}\right), \quad D \neq i \tag{2.5.5}$$

The dissimilarity measures evaluated in this study are listed below.

Aitchison $$[\sum_{k=1}^{s}(\log(\frac{x_{ik}}{g(\boldsymbol{x}_i)}) - log(\frac{x_{jk}}{g(\boldsymbol{x}_j)}))^2]^{\frac{1}{2}} \quad (2.5.6)$$

Angular Separation (arccos) $$\arccos\frac{(\sum_{k=1}^{s} x_{ik}x_{jk})}{\sqrt{(\sum_{k=1}^{s} x_{ik}^2 \sum_{k=1}^{s} x_{jk}^2)}} \quad (2.5.7)$$

Bhattacharyya (arccos) $$\arccos(\sum_{k=1}^{s}\sqrt{x_{ik}}\sqrt{x_{jk}}) \quad (2.5.8)$$

Bhattacharyya (log) $$-\log(\sum_{k=1}^{s}\sqrt{x_{ik}}\sqrt{x_{jk}}) \quad (2.5.9)$$

City Block $$\sum_{k=1}^{s}|x_{ik} - x_{jk}| \quad (2.5.10)$$

Divergence $$\frac{1}{s}\sum_{k=1}^{s}\frac{(x_{ik} - x_{jk})^2}{(x_{ik} + x_{jk})^2} \quad (2.5.11)$$

Euclidean $$[\sum_{k=1}^{s}(x_{ik} - x_{jk})^2]^{\frac{1}{2}} \quad (2.5.12)$$

The angular separation (2.5.6) and Bhattacharyya distances (2.5.8 and 2.5.9) are related to correlation. Cha (2007) classifies angular separation (called cosine coefficient in his paper) as part of the inner product family. It is the normalized inner product, and is called the cosine coefficient because it measures the angle between two vectors.

The Euclidean distance (2.5.12), city block distance (2.5.10) and divergence (2.5.11) are all additive dissimilarities. Euclidean and city block distances are part of the Minkowski family of dissimilarities $[\sum_{k=1}^{s}(x_{ik} - x_{jk})^p]^{\frac{1}{p}}$ with $p = 2$ and $p = 1$ respectively.

Euclidean distance places the same weight on any two points 'equally far' apart in the space, whereas Divergence increases the weight for distances between points with small values. Similarly, Aitchison is based on a log transformation which affect larger values more than smaller values, bringing larger values closer to each other, and therefore induce smaller distance values than Euclidean distance. Therefore, the distance measures selected will have an effect on the placement of the design points due to the difference in how the distances are calculated. For example, Divergence yields a natural scaling, whereas Aitchison is a logarithmic transformation. Different practical situations relating to the constraintness of the mixture space might require the use of a different measure.

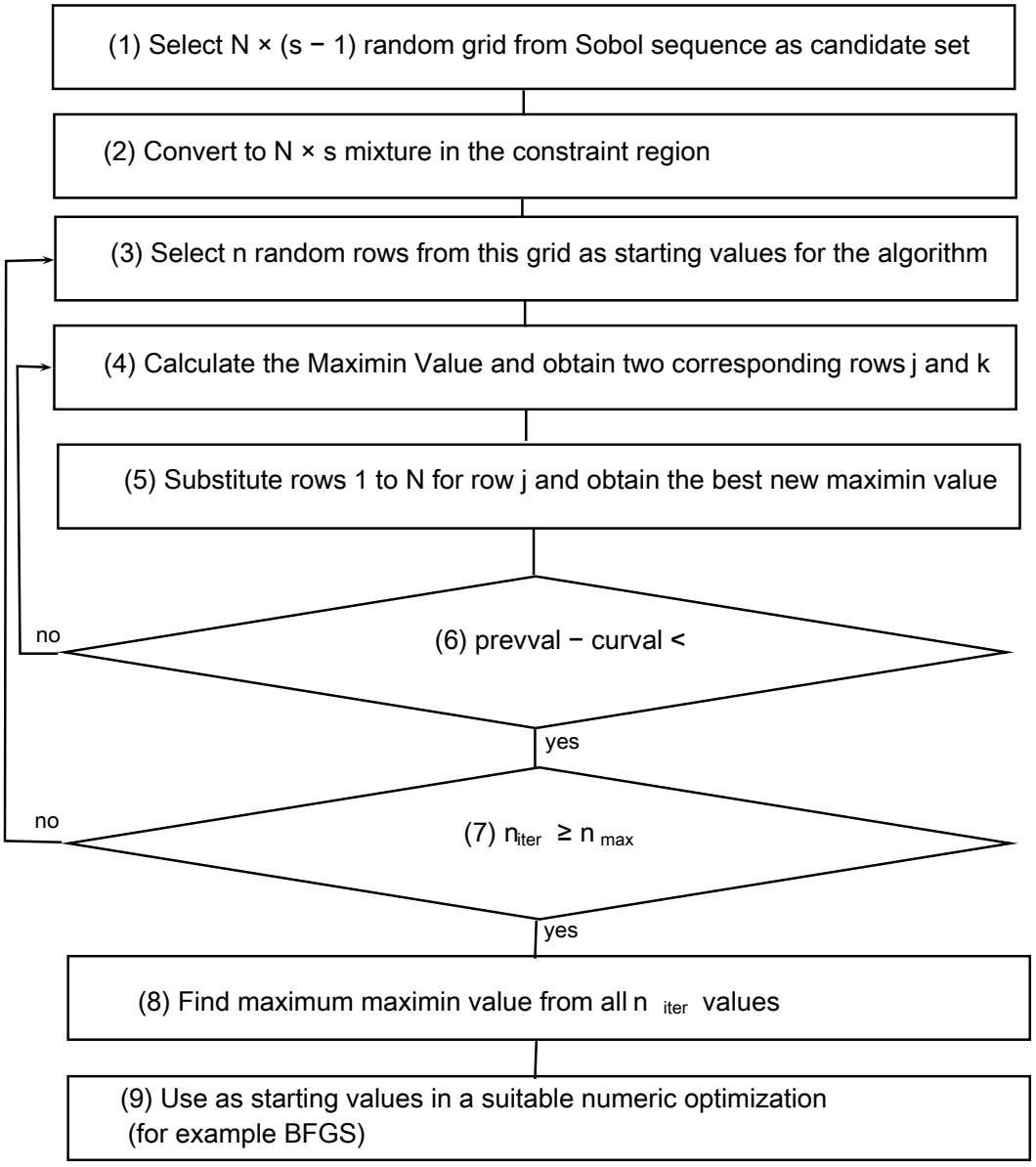


Figure 2.24: Algorithmic Overview of Optimization Methodology

2.5.1.2 Optimization Methodology

Creating space-filling designs is a very challenging optimization problem. In the case of mixture experiments the space is often highly constrained in addition to the nonlinear nature of the optimization criterion. In order to compare different designs it is important that the optimization strategy used is appropriate for each design in order to create a level playing field i.e., designs should not be artificially penalized due to inappropriate optimization methodologies.

Coetzer *et al.* (2012) proposed the optimization philosophy illustrated in Figure 2.24. First a large grid of dimension N (where N is some large number)

by $s-1$ is created as a candidate set using a Sobol sequence (Bratley and Fox, 1988), then this grid is transformed to a $N \times s$ mixture candidate set by using the methodology discussed in Borkowski and Piepel (2009). An initial design is generated by selecting n distinct points from the grid of N points. The maximin values are then calculated, and the two points with minimum distance, say i and j, returned with the minimum value calculated. Each of the N points in the candidate set is then substituted for point i in the design until a point with a larger minimum value is found. This point is then substituted for point i. This process is repeated from step 4 until the increase in the minimum value is below a very small value (ϵ). A new n point initial design is then created (step 3) and the process is repeated from step 3 for a predetermined number of iterations n_{iter}. In step 8 the design with the maximum of the maximin values is selected, and then used as initial values for the algorithm discussed in Stinstra *et al.* (2003). The algorithm was however updated to only substitute the points corresponding to the minimum value for computational efficiency.

As discussed in Coetzer *et al.* (2012) the maximin design criterion has some attractive properties that can be exploited in the optimization steps. Specifically the optimization criterion is only concerned with the two design points that lie closest to each other in the design space. Any change in the movement of the other $n-2$ design points will not affect the criterion at all. This has both positive and negative consequences. On the positive side only the two points closest to each other need to be evaluated for each iteration of the optimisation algorithm. This has obvious consequences for optimization efficiency. This approach was implemented by changing the objective function to not only calculate the maximin value, but also to return the two points being evaluated.

However, even though the space-fillingness of the other $n-2$ points may be suboptimal this has no effect on the maximin value. In addition, if two or more sets of points lie on the same minimum distance from each other the algorithm can get stuck. In this case it is not possible to increase the maximin value by only changing one point (as most optimization algorithms do). Coetzer *et al.* (2012) proposed that the number of points at the minimum distance is determined and a small value (ϵ) times that number subtracted from the minimum distance to compensate for this effect.

2.5.1.3 Design Evaluation Criteria

To effectively compare different space-filling designs an evaluation criterion must be specified. The Kullback-Leibler Information and the Eigenvalue Properties of the designs are considered.

Kullback-Leibler Information The Kullback-Leibler (KL) information statistic measures the difference between two density functions f and g (Jourdan

and Franco, 2009), i.e.

$$I(f,g) = \int f(\boldsymbol{x}) \log\left(\frac{f(\boldsymbol{x})}{g(\boldsymbol{x})}\right) dx \tag{2.5.13}$$

If g is the uniform density function, then the KL information becomes

$$I(f,g) = \int f(\boldsymbol{x}) \log f(\boldsymbol{x}) dx = -H[f] \tag{2.5.14}$$

where $H[f]$ denotes the Shannon entropy. Therefore, minimizing the KL information statistic amounts to maximizing the entropy. Entropy is defined (Shewry and Wynn, 1987) as:

$$\begin{aligned} H[f] &= -E_x(\log f(\boldsymbol{x})) \\ &= -\int f(\boldsymbol{x}) \log f(\boldsymbol{x}) dx \end{aligned} \tag{2.5.15}$$

which can be approximated by:

$$\hat{H}[f] = -\frac{1}{N}\sum_{i=1}^{N} \log f(\boldsymbol{x}_i) \tag{2.5.16}$$

where N is some large number.

For mixture data the Dirichlet distribution can be used to model the data (Aitchison, 1986). The density function for the Dirichlet class, for an observation is (Naryanan, 1991):

$$f(\mathbf{x}) = \frac{\Gamma(\alpha_1 + \ldots + \alpha_s)}{\Gamma(\alpha_1)\ldots\Gamma(\alpha_s)} x_1^{\alpha_1 - 1} \ldots x_s^{\alpha_s - 1} \tag{2.5.17}$$

where $\sum_{i=1}^{s} x_i = 1$.

The log-likelihood of (2.5.17) is:

$$\log L = n\left\{\log\Gamma\left(\sum_{j=1}^{s}\alpha_j\right) - \sum_{j=1}^{s}\log\Gamma(\alpha_j)\right\} + \sum_{j=1}^{s}\left\{(\alpha_j - 1)\sum_{i=1}^{n}\log x_{ij}\right\} \tag{2.5.18}$$

Therefore, we calculate the entropy (2.5.16) and compare the designs generated with the Kullback-Leibler information over the different dissimilarity measures.

Eigenvalue Criteria The eigenvalue properties of the $n \times s$ design matrix $\boldsymbol{X}$ could be used in a design evaluation criterion. Specifically, from the eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$ the following properties (or some combination) are investigated as design evaluation criteria:

Table 2.12: Table of generic mixture properties and ranges

Component	Minimum	Maximum
x_1	0.1	0.8
x_2	0.1	0.8
x_3	0.1	0.8

- The condition number $\frac{\lambda_{max}}{\lambda_{min}}$ for the design. A value closer to 1 is desirable as it indicates approximately equal eigenvalues.
- $\frac{\lambda_{max}}{\sum_{i=1}^{s}\lambda_i}$ for each design. Note: if each $\lambda_i = \lambda$, then $\frac{\lambda}{\sum_{i=1}^{s}\lambda_i} = \frac{\lambda}{s\lambda} = \frac{1}{s}$, which is the best design. Therefore, each dimension is represented in the design equally well/efficient.

The designs generated according to the different dissimilarity measures are compared with respect to these eigenvalue properties.

2.5.1.4 Results

Coetzer *et al.* (2012) applied the optimisation algorithm to the generic mixture in Table 2.12 for each of the dissimilarity criteria. The design evaluation criteria were calculated for each of the resulting maximin designs. The results are provided in Table 2.13. From Table 2.13 it can be concluded that the Divergence dissimilarity measure performs the best for three of the five criteria. Therefore, the Divergence dissimilarity measure was proposed as a dissimilarity measure for creating space-filling designs for mixture experiments.

Given that the designs consist of three components it is possible to use a ternary plot to visualise the final designs. Care must however be taken in interpreting the ternary plot, as Euclidean distance is used to create the plot. However, it is still insightful to visualise the designs. The plots for the optimal maximin design using Aitchison distance and Divergence as dissimilarity criteria are shown in Figures 2.25 and 2.26 respectively. Comparing these plots with the plot for the optimal maximin design using Euclidean distance as dissimilarity (Figure 2.27), it is clear that care should be taken to visually compare designs in the Euclidean space. Visual inspection would indicate that the Euclidean distance yields the best space-filling design although the evaluation criteria indicates differently.

2.5.2 Slagging Model

For the slag prediction model Divergence was used as dissimilarity measure, as it performed the best on the Kullback-Leibler Information measure as well as the eigenvalue properties of the untransformed data. Divergence also converged orders of magnitude faster than the Aitchison distance measure, which,

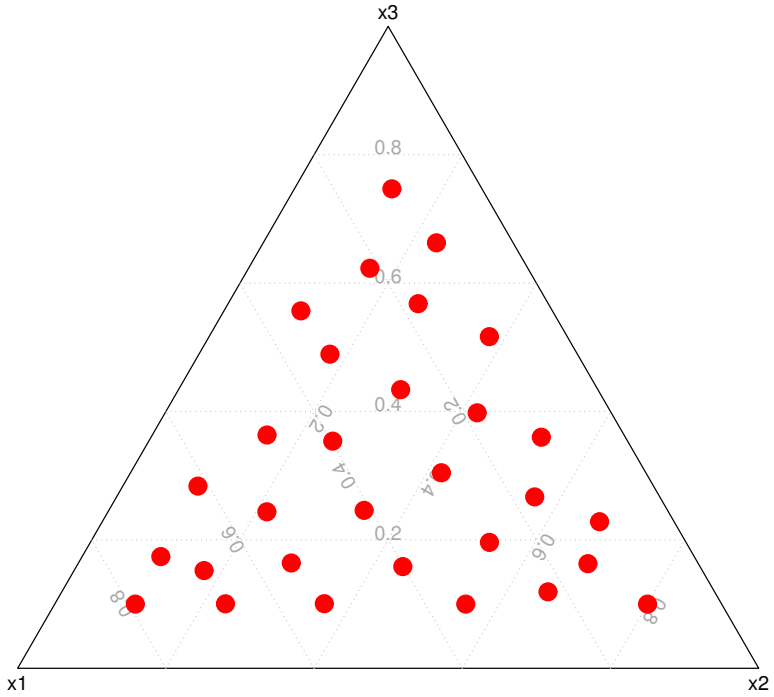


Figure 2.25: Ternary plot of the optimal Minimax design using Aitchison distance as dissimilarity criterion

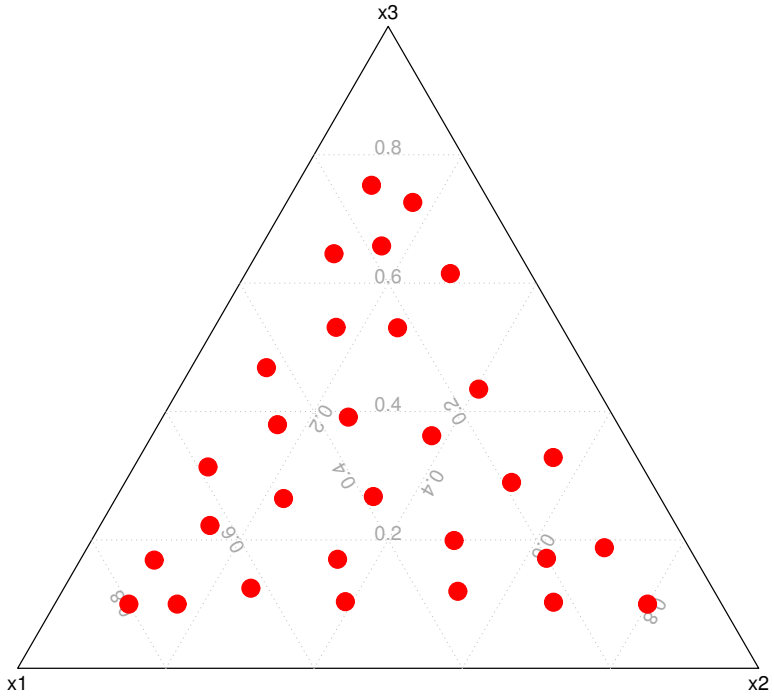


Figure 2.26: Ternary plot of the optimal Minimax design using Divergence as dissimilarity criterion

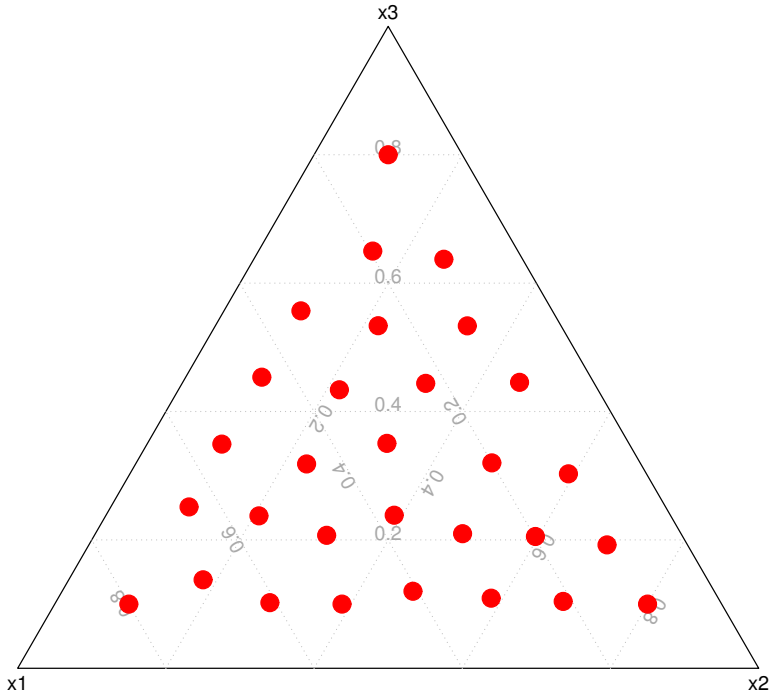


Figure 2.27: Ternary plot of the optimal Minimax design using Euclidean distance as dissimilarity criterion

Table 2.13: Comparison of space filling designs and models for different properties for the generic three variable case.

Transformation	**Criterion**	**Aitchison**	Bhattacharyya		**City-block**	**Divergence**	**Euclidean**	**Angular Separation**
			arccos	**log**				**arccos**
Raw	$I(f,g)$	0.923	0.913	0.924	0.971	0.890	0.916	1.001
ALR	$\frac{\lambda_{max}}{\lambda_{min}}$	2.670	2.910	3.350	3.350	2.940	2.850	3.160
ALR	$\frac{\lambda_{max}}{\sum_{i=1}^{s}\lambda_i}$	0.513	0.523	0.544	0.530	0.526	0.524	0.516
Raw	$\frac{\lambda_{max}}{\lambda_{min}}$	6.520	5.680	6.120	6.900	5.510	6.230	6.960
Raw	$\frac{\lambda_{max}}{\sum_{i=1}^{s}\lambda_i}$	0.741	0.737	0.744	0.759	0.727	0.746	0.771

especially for the higher dimensional problems, is a very important property. The ranges for the ash compositions are tabulated in Table 2.14. It can be observed from Table 2.14 that the design space is highly constrained.

Generally the rule of thumb is to choose $n = s \times 10$, but due to additional constraints the maximum number of model runs was specified as 74. N was chosen as 50000 with 10000 iterations of the optimization algorithm. The large amount of iterations was performed because the design space is highly constrained and irregular, and the problem has 10 dimensions.

The optimal maximin design points are illustrated graphically in Figure 2.28 using the scaled component values (Borkowski and Piepel, 2009). This

Table 2.14: Ash composition ranges

	Minimum	Maximum
V1	0.4030	0.6516
V2	0.1740	0.3200
V3	0.0110	0.1273
V4	0.0060	0.0097
V5	0.0120	0.0162
V6	0.0190	0.1821
V7	0.0180	0.0248
V8	0.0070	0.0108
V9	0.0040	0.0043
V10	0.0420	0.0723

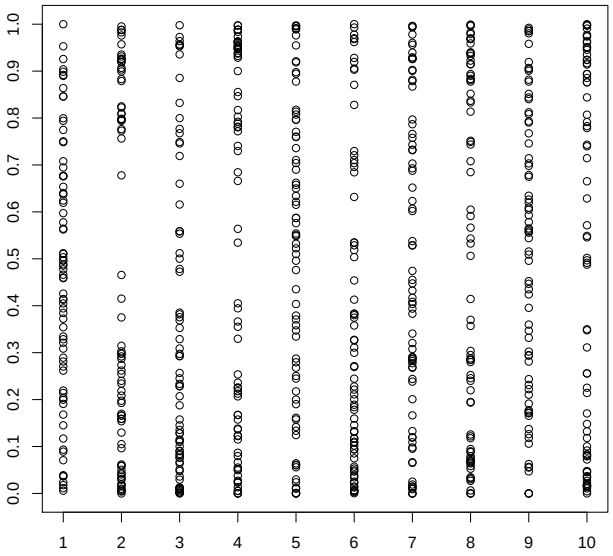


Figure 2.28: Scaled Component Values plot of maximin design for ash composition

plot depicts all the components individually scaled to their individual ranges, i.e each $x_{ij} \in \boldsymbol{X}$ is scaled to x^*_{ij} by (2.5.19).

$$x^*_{ij} = \frac{x_{ij} - L_j}{U_j - L_j} \tag{2.5.19}$$

Although this plot does not give any information about the design structure in higher dimensions it does give an indication of the spread of design points for each individual component. From Figure 2.28 it can be observed that the

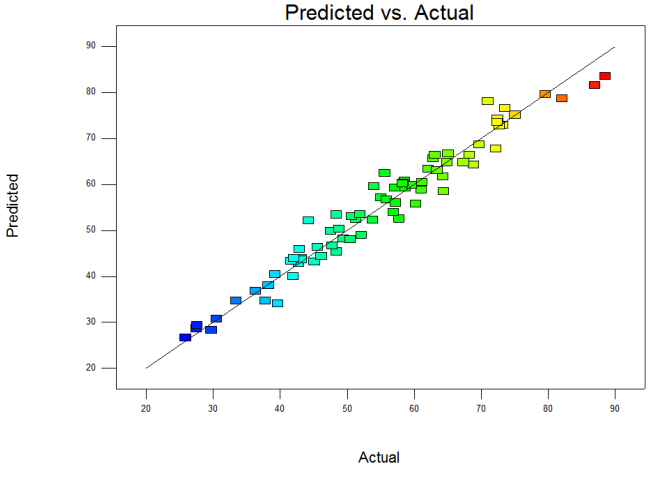


Figure 2.29: Actual versus predicted plot for slag prediction model

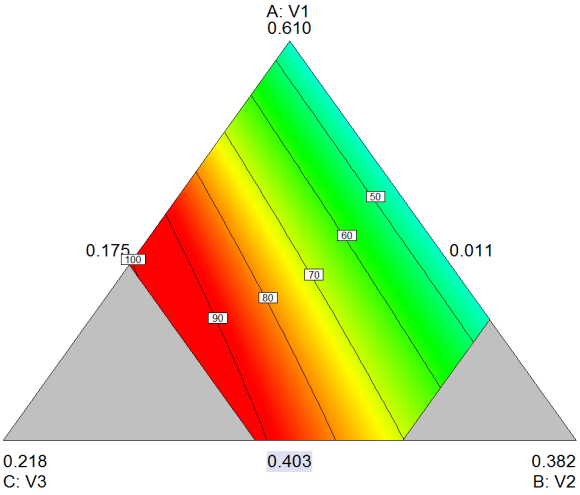


Figure 2.30: Ternary plot the predicted slag percent as a function of the first three ash composition variables

design points are reasonably well distributed across all the dimensions.

The FactSage model was used to obtain the slag predictions for the 74 design runs. The data were imported into the Design Expert software (Stat-Ease, 2002) and a quadratic model was fitted. An R-Squared value of 0.9124 was obtained. Figure 2.29 depicts the actual versus predicted values for the fitted model. Both the high adjusted R-Squared value and the excellent comparison of the predicted to the actual values indicate that the model can be used as a surrogate for the FactSage model. Figure 2.30 shows the predicted trend in slag formation as a function of the first three ash composition variables. The linear increase in slag formation with an increase in the percentage of variable three in the composition is clearly visible. The model results highlight the

quality of the optimal maximin design generated with the Divergence dissimilarity criterion and the optimisation strategy presented in Section 2.5.1.2. The surrogate model is utilised for the real-time prediction of the slagging of the ash in the MSPEM™ application.

2.6 Conclusions

In this chapter various statistical and Data Science techniques were presented to close the gap between the data that are available, and the requirements with regards to real time coal quality information of the Coal Value Chain. Specifically, Excel files from SGS Laboratories, XRF Data, Material Movement Data, Stockpile Information files and the PI DCS system were captured, cleaned and stored in a standardised format for real time on-line processing.

All the data sources in isolation, although valuable, do not add the level of insight that can be extracted from combining them. A stacker simulation model that was developed and implemented to predict the properties of the heaps in all the stack yards utilising the XRF data in combination with the data form the SGS laboratories, the material movement files and the stockpile information files, was discussed. The stacker simulation model allows for the prediction of coal properties (including but not exclusive to ash) over the length of the heap, as well as the average ash percentage and standard error of the ash percentage for the heap. This information is used to do blend planning for the week, and is compared to the lab analysis data for validation purposes.

The output from the stacker simulation model was combined with the data from the material movement files, and the reclaimer data from the PI DCS system to calculate the reclaimed ash and standard deviation of the ash going to the gasification plant. This information is now used to develop a reclaiming strategy to minimise the impact of the heaps with unfavourable ash properties on the gasification factory. The data extracted from the various data sources at the SCS facilities have therefore been converted into valuable information that can be used for making informed decisions.

In the last section a design and analysis of computer experiments strategy has been applied to an existing FactSage computer model for predicting slag formation of the ash. The input to the slagging model consists of ash elemental properties that can be obtained from the XRF analyser. Due to the compositional nature of the ash elements data, a space filling design appropriate for mixture experiments was required. Various dissimilarity measures appropriate to mixture experiments were compared using two design comparison measures. In addition, an optimisation strategy was presented for obtaining maximin designs. The resulting optimal dissimilarity measure (Divergence) and optimisation strategy were applied to obtain a mixture design. The model runs were obtained for the design points, and a quadratic model was fitted to the data. The predicted results from the quadratic model compared very well to the

actual results, and it was therefore concluded that the design was appropriate for the model. This model can now be used to do real time on-line prediction of the slag formation of the ash.

In summary, in this chapter the real time coal quality analyses from an XRF analyser were summarised and integrated with various data sources from the Coal Supply Facility to provide information on the coal quality of each mine. In addition, simulation models were developed to generate information on the coal quality of each heap and the quality of the reclaimed coal sent to gasification. Finally, the novel application of distance measures other than Euclidean measures was introduced for space filling design for computer experiments in mixture variables.

In the next chapter a monitoring strategy for the Sasol Coal Gasification (SCG) plant will be developed and presented.

Chapter 3

Coal Gasification

The Sasol Coal Gasification (SCG) plant is a highly complex facility. The system consists of two separate facilities known as Gasification West and Gasification East. Each facility consists of four trains, each containing between 10 and 11 gasifiers (see Figure 3.1). Each gasifier is equipped with instrumentation which records online performance data on the gasifiers. The coal from the SCS facility discussed in Chapter 2 comprises the main feedstock to the SCG plant. The coal qualities therefore have a direct impact on the performance of the SCG plant. Monitoring and comparing the performance of the 84 gasifiers are of utmost importance as the units supply the feedstock for the downstream units in the coal to liquids facility.

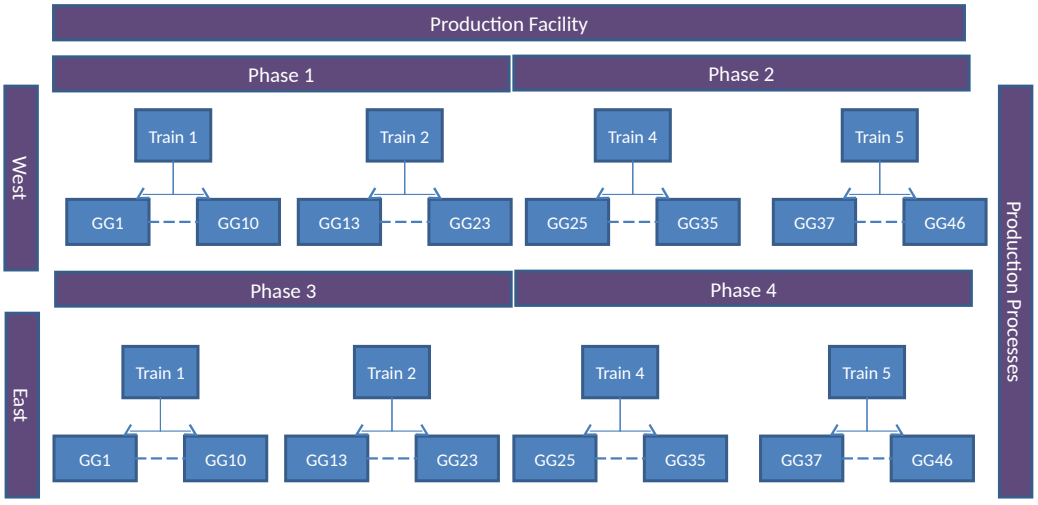


Figure 3.1: Gasification Overview

3.1 Coal Gasification Monitoring

The coal gasification plant is a unique facility as it is the largest coal to liquids facility in the world. A need was identified for a custom product that can facilitate the monitoring of the gasification facility specifically for engineering managers. This product should utilize the real time data, and convert it to information on the different layers of the production facility (i.e. Factory, Side of Factory, Production Train, Gasifier). Even though the conversion of data into the appropriate information may be technically involved, the results should be intuitive, and easy to interpret.

The developed product discussed in this chapter entails an efficient multivariate process monitoring methodology for those process variables that govern gasifier performance. These variables can be divided into three groups, production, utility and stability variables. From a monitoring perspective there are two different but complimentary strategies that can be followed for process monitoring. The first strategy is a *process driven (fundamental) approach* where information from the subject matter experts is utilized to specify for example the optimal operating ranges for the variables. A fundamental approach will be discussed in Chapter 4. Alternatively a *data driven (empirical) approach* can be followed where historical data are used to specify the optimal operating ranges. These two approaches do overlap, and in Chapter 4 an integrated approach will be employed to develop a monitoring strategy.

Real time monitoring of the gasifier process variables is the primary objective. The process data are captured in real time on a distributed control system (DCS), and may be captured at different time intervals for the different variables. From a data perspective some of the problems that need to be addressed entail the selection of an appropriate aggregation window, as well as an appropriate aggregation method for each process variable. The aggregation methods will be discussed in detail in Chapter 5.

As all ten gasifiers on one train receive the same coal, and are managed by the same operator it is expected that the performance will be similar. Any deviation of performance of a gasifier from the mean performance of all the gasifiers on the train can be an indication of mechanical problems. Therefore, it is important to evaluate the differences between the gasifiers on a train. This is a longer term approach, but should still be available in real time. In Section 3.4 Canonical Variate Analysis (CVA) biplots will be proposed as an approach to this problem.

In the current application the development of the empirical real time monitoring methodology follows three consecutive steps. Therefore, this chapter is divided into the following three sections:

- Section 3.2 - The selection of the reference set.
- Section 3.3 - The utilisation of the reference set for multivariate process monitoring using the PCA biplot.

- Section 3.4 - Using the CVA Biplot for monitoring.

Each section focuses on the specific topic, and to facilitate the flow of the discussion a methodology will sometimes be introduced and utilised before the relevant theoretical background is provided. For example, the PCA biplot is utilised in the section on reference set selection, but will only be discussed in detail in the following section on multivariate process monitoring using the PCA biplot.

The first aspect to be addressed in this chapter is to find the optimal operating window and period of stable operation. Therefore, the expected behavior of the process needs to be specified. This is normally achieved by selecting a reference set of data from an historical time period where the process was running stable and within expectation. The correct selection of the reference data set is crucial to the success of real time process monitoring. Generalized Orthogonal Procrustes Analysis (GOPA) (Gower and Dijksterhuis, 2004) will be employed in this study for selecting the optimal reference set for the multivariate monitoring of the multiple identical production processes (i.e. gasifiers).

3.2 Reference set Selection

3.2.1 Introduction

Coetzer *et al.* (2014) discussed reference set selection for a reduced version of the gasification production process. In this section the same approach will be employed but expanded to the full production facility.

Process monitoring and more specifically multivariate statistical process control has received much attention in the statistical and engineering literature (Aldrich *et al.*, 2004; Ferrer, 2014; Kourti and MacGregor, 1995; MacGregor, 1997; MacGregor and Kourti, 1995; Sparks *et al.*, 1997). The fundamental approach of the majority of the multivariate process monitoring procedures is to first specify a historical reference set that is within statistical control. Multivariate analytical techniques such as Principal Component Analysis (PCA) are then employed to project process variables onto a lower dimensional space where they can be jointly monitored given the in-control reference set. Ferrer (2014) calls this approach to multivariate statistical process control Latent Structures-Based multivariate statistical process control (LSb-MSPC). According to Ferrer (2014), the LSb-MSPC scheme is carried out in two phases:

- *Phase I (model building).* The main goal of Phase I is to model the in-control process performance based on a set of in-control reference data. If the reference data are not available, it is extracted from the historical database in an iterative process.

- *Phase II (model exploitation).* Once the reference PCA model and the control limits for the multivariate control charts are obtained, new process observations can be monitored on-line.

Throughout this study it is assumed that a reference set is required for monitoring whether the process is within statistical control and to detect out-of-control or deviations from expected performance i.e., the reference set is considered as the target for the process. The use of the PCA biplot as a monitoring graph has been discussed at length by Aldrich *et al.* (2004), and more specifically the use of the biplot as a dynamic tool, which is updated in real time, has been discussed by Sparks *et al.* (1997). The PCA biplot as a monitoring tool will be revisited in Section 3.3.3.

Current literature is mostly focused on multivariate statistical monitoring of many process variables simultaneously for a single production process. The selection of a single reference set that is within statistical control or conforms to some specified accepted performance measure(s) for multiple identical production processes simultaneously has not been discussed previously. In this study a methodology for selecting a single reference set for multivariate process monitoring across multiple identical production processes is presented.

Real time data are captured on more than 10 process variables for monitoring the performance of each gasifier, with the main output from the individual gasifiers the amount of raw gas (km^3n/h) produced. The gasifiers are referred to as the production processes. The production processes are grouped into a number of production trains e.g., 10 production processes per train (see Figure 3.1). All the trains receive the same feedstock, but each train has a different operator. Therefore, differences between trains and production processes may be due to operator, mechanical degradation or other process variable deviations.

In this study the problem of selecting one production train as a reference set that allows for efficient multivariate process monitoring of all the production processes is considered. The motivation behind the selection of one train is that the whole production facility should ideally be operated similarly, and deviations in performance of the production processes are evaluated relative to the same reference of expected or optimal performance. As the production trains (and therefore the production processes on the trains) receive similar feedstock together with mechanically identical production processes, the major cause of differences in performance over trains must be related to operating protocol. Selecting the optimal train as reference set would therefore ensure that the operators on all the trains target the same optimal operating protocol.

The coal gasification process is a continuous process and the performance of the reactors is monitored in real time. However, the performance of the production processes may change from period to period due to planned or unexpected changes in the feed. In Coetzer *et al.* (2014) a week was used as base time unit as the feedstock blending is updated weekly according to feed

availability, and other drivers (See Chapter 2). The feedstock could therefore potentially change on a weekly basis. The larger production facility is however governed by a monthly budget plan. Therefore a base time unit for reference set selection should be a month of data. In addition, the optimal combination of a specified number of months to employ for the selection of the optimal train as the reference set are of interest.

A methodology using Generalised Orthogonal Procrustes Analysis (GOPA) (Gower and Dijksterhuis, 2004) for selecting the optimal reference set for the multivariate monitoring of the multiple identical production processes is presented. More specifically, GOPA is applied to select the optimal combination of a specified number of months as well as the optimal production train. Procrustes analysis and the more generalized GOPA have not been applied previously for reference set selection within the context of multivariate process monitoring. Although it is not the case here, it is noted that this procedure allows for different variables to be measured at the various production processes.

3.2.2 Problem setting for reference set selection

Consider Figure 3.1, which represents a flowsheet of the production facility under study. Specifically, the facility under investigation consists of two separate but identical production facilities (East and West), with each containing four trains with either 10 or 11 gasifiers (production processes) per train for a total of 84 production processes (42 West and 42 East). Note that the numbering of the trains and processes is not consecutive as there were numbers omitted for future expansion. Note in this study the gasifiers will be abbreviated by GG, with numbers as references to indicate the specific gasifier, for example GG01 East for gasifier 1 on the Eastern Factory. In the current study, time weighted fifteen minute average data were continuously captured for a 35 month period for 84 production processes for eleven process variables. Note that due to confidentiality constraints the actual months were encoded as M01 to M35. The process variables include production volume, utility consumption such as oxygen and steam consumption and other stability measures on the reactors. Table 3.1 provides a description of the types of variables used in the study. Note that the variables are indicated by neutral labels due to confidentiality restrictions imposed by the company under consideration. In addition, the production volume variable was removed from this study due to measurement inconsistencies in the data. This posed no constraint as the main driver of this study was stability.

Canonical variate analysis (CVA) biplots are employed to groups of data to maximize the between group relative to within group variance (Gower *et al.*, 2011). A detailed discussion of CVA biplots will be provided in Section 3.4. To illustrate the complexity of the industrial problem, a CVA biplot was constructed to visualize the differences between the different production processes

Table 3.1: Variable types

Variable	Type
U1	Utility
U2	Utility
U3	Utility
S1	Stability
S2	Stability
S3	Stability
S4	Stability
S5	Stability
S6	Stability
S7	Stability
U4	Utility

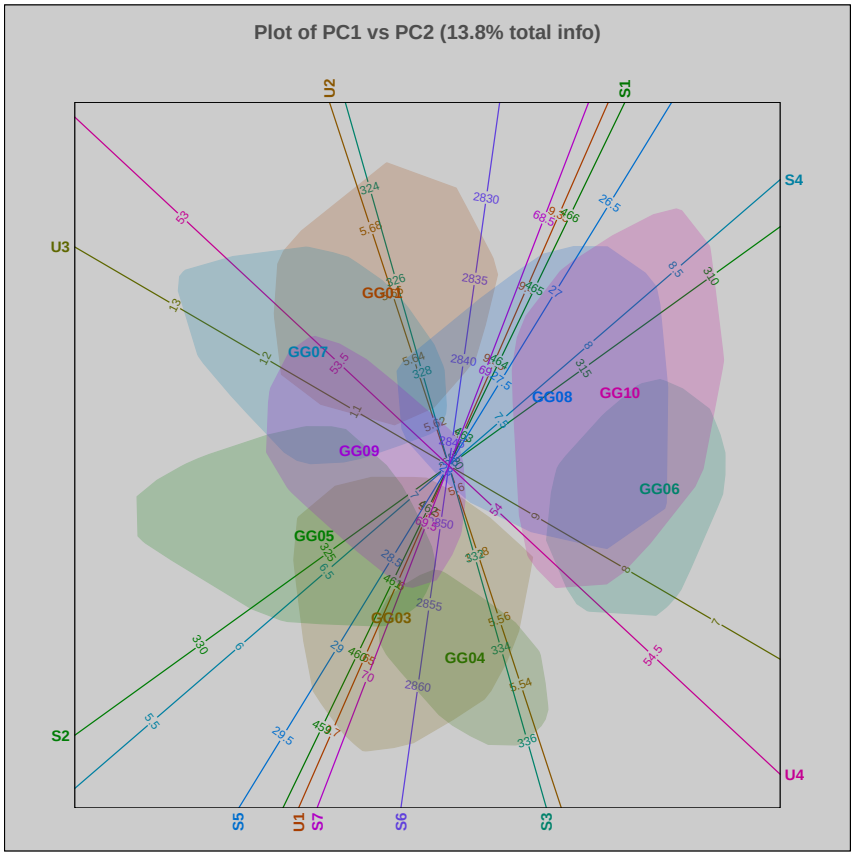
for some selected weeks. Figure 3.2 depicts the CVA biplot for all four trains for the Eastern factory over a one week period. Ninety percent (90%) alpha bags are added to the display to quantify the within group variability for each group (Gardner-Lubbe *et al.*, 2008). From Figure 3.2a, it is observed that on average the gasifiers on the train perform similar, while from Figure 3.2b it follows that GG17 and GG23 are projecting higher on variables U1, U4 and S7 and lower on variable S4 compared to the other gasifiers. Similarly GG15 projects higher on the variables U1, U4 and S7 and higher on variable S4 compared to the other gasifiers. It can be observed from Figure 3.2c that Train Four was operating more stable. From Figure 3.2d it is clear that GG44 projects lower on stability variables S2 and S3 compared to the other gasifiers. The differences between the trains may be due to different operators, but the differences between the reactors on the same train may be due to the specific process variable deviations or mechanical wear and tear.

The use of CVA biplots for the longer term monitoring of the gasifiers on the trains as well as the interpretation of the CVA biplots will be discussed in detail in Section 3.4. Specifically, the effect of axis predictivity and the choice of dimensions on the interpretation will be discussed in Section 3.4.4. The CVA biplots in Figure 3.2 illustrate the differences in performance between the production processes across different trains. Similarly the difference for individual trains over different months can be observed. Therefore, the selection of the optimal combination of months and train in Phase I will provide the ability to monitor deviations in performance of the production processes relative to the same reference of expected or optimal feed. The problem reflects the difficulties with the implementation of a monitoring methodology on the actual production facility, where a reference set needs to be selected for the simultaneous monitoring of all the trains or processes which may consist of up to 11 production processes per train. GOPA (Gower and Dijksterhuis, 2004) is applied for selecting the reference set of the optimal combination of

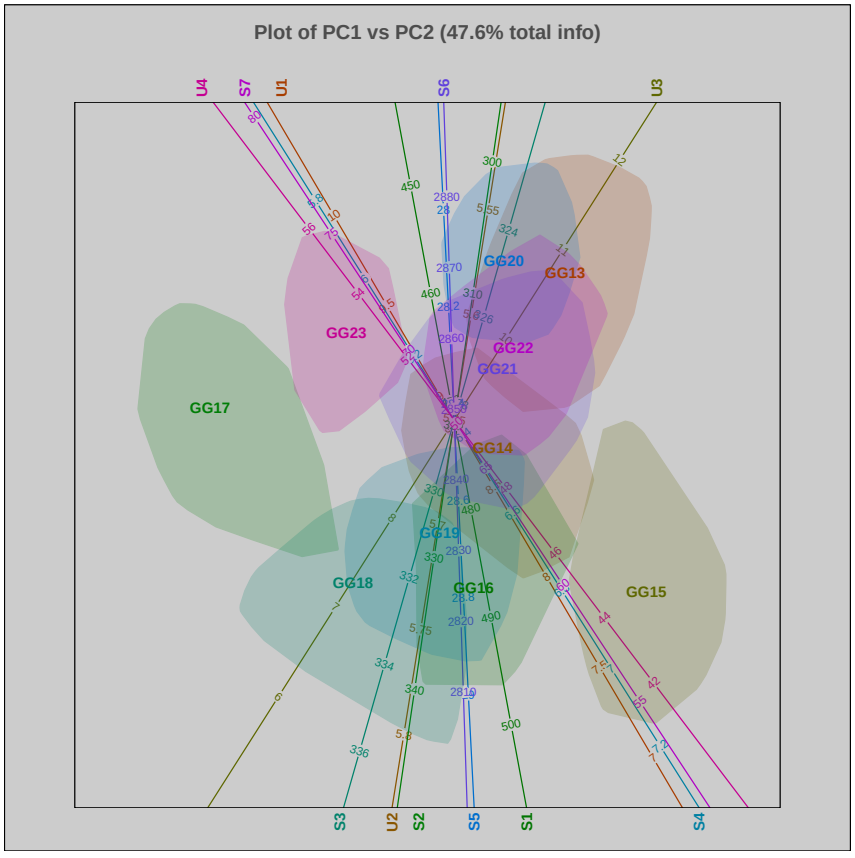
weeks and the production train.

In the current problem where all the production processes have the same process variables, it might be argued that it is more appropriate to apply methods which are optimal for comparing the multivariate means. However, as demonstrated above, methods such as CVA cannot be used to determine the optimal train for the reference set since the within variance-covariance structure of the data is as important as the differences between the means. For example, it could happen that although the multivariate means of the process variables are different, the within variance-covariance structure present in the data is the same or vice versa. Also, a flow meter may have a consistent error, but the relative differences about this offset is measured correctly. Therefore, instead of performing an univariate comparison of the structure about the means, Procrustes analysis can be employed to remove all the superfluous differences between the reactors.

No evidence was found in the statistical and engineering literature that GOPA had been applied previously in selecting a reference set for the multivariate process monitoring of multiple production processes.

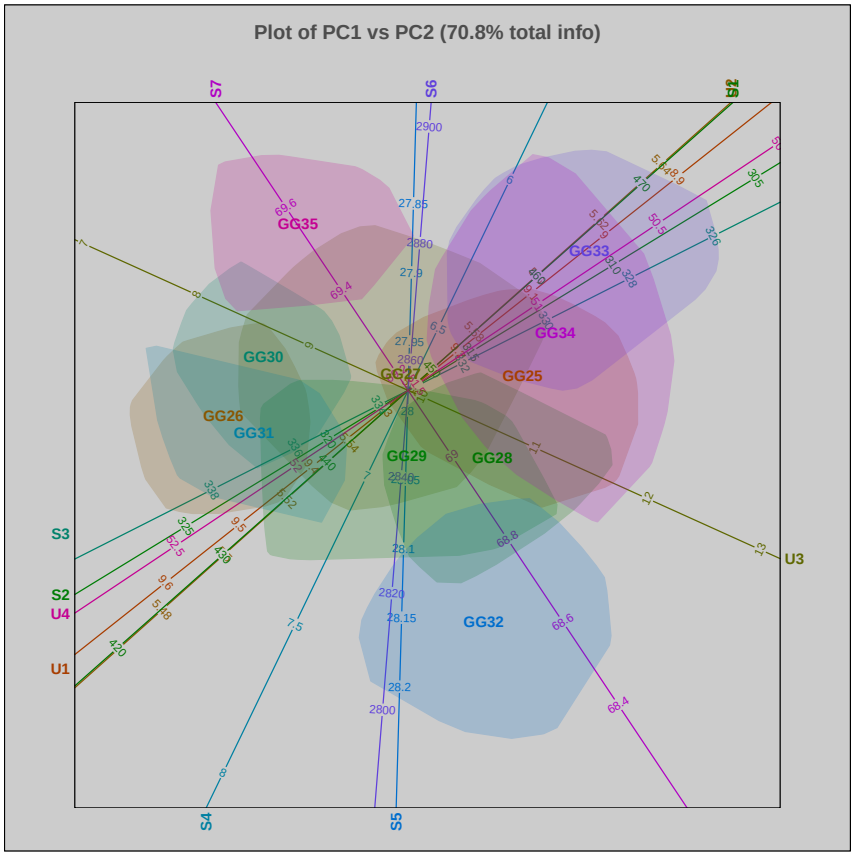


(a) CVA Biplot for Train 1 East

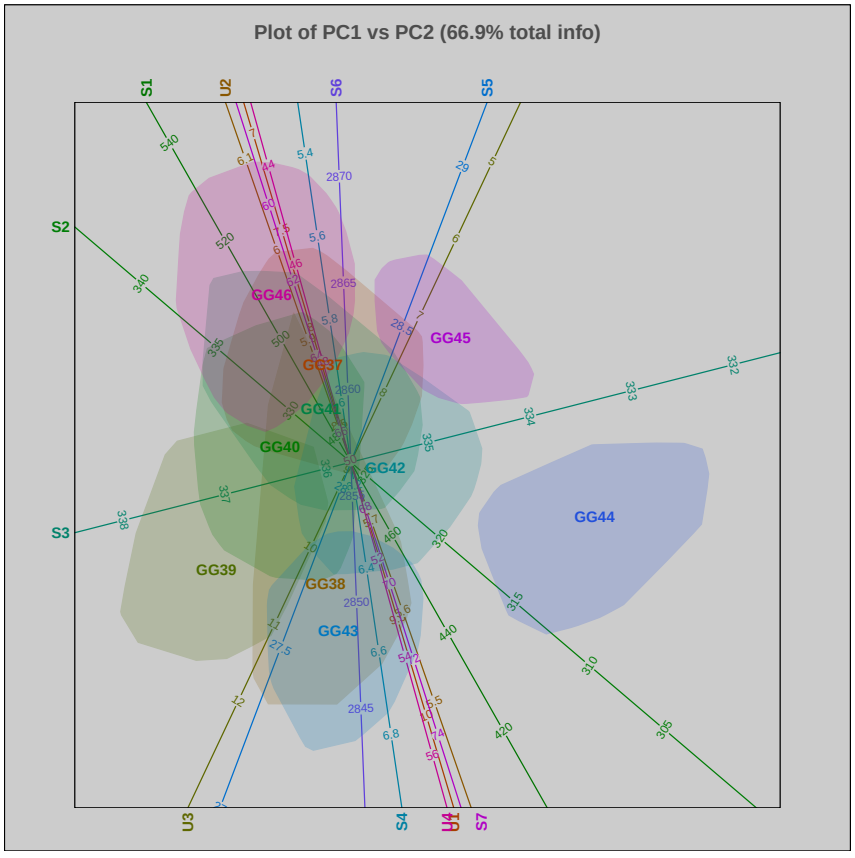


(b) CVA Biplot for Train 2 East

Figure 3.2: CVA Biplots for Eastern Factory for a one week period



(c) CVA Biplot for Train 4 East



(d) CVA Biplot for Train 5 East

Figure 3.2: CVA Biplots for Eastern Factory for a one week period continued

3.2.3 A Brief Introduction to Procrustes Analysis

For a clear understanding of the proposed method a brief overview of the theory underlying GOPA is provided. The simplest algebraic form of the Procrustes problem is seeking a matrix $\boldsymbol{B}$ which minimizes the sum of squares

$$\|\boldsymbol{X}_1\boldsymbol{B} - \boldsymbol{X}_2\|^2 \tag{3.2.1}$$

over $\boldsymbol{B}$ where $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are the two configurations that need to be matched, and $\boldsymbol{B}$ is the matrix that operates on $\boldsymbol{X}_1$ to minimize (3.2.1). In (3.2.1) $\|\boldsymbol{X}\|^2$ is defined as the squared Euclidean norm trace$(\boldsymbol{X}^T\boldsymbol{X})$, the sum of squares of the elements of $\boldsymbol{X}$. In this study the orthogonal Procrustes problem is discussed. That is, $\boldsymbol{B}$ is constrained to be orthogonal. Constraining $\boldsymbol{B}$ to be orthogonal allows for an analytical solution to the Procrustes problem (Gower and Dijksterhuis, 2004). The sum of squares $\|\boldsymbol{X}_1\boldsymbol{B} - \boldsymbol{X}_2\|^2$ can be expanded as

$$\|\boldsymbol{X}_1\boldsymbol{B} - \boldsymbol{X}_2\|^2 = \text{trace}(\boldsymbol{X}_1^T\boldsymbol{X}_1 + \boldsymbol{X}_2^T\boldsymbol{X}_2) - 2\text{trace}(\boldsymbol{X}_2^T\boldsymbol{X}_1\boldsymbol{B}). \tag{3.2.2}$$

The first term on the right hand side of (3.2.2) does not contain $\mathbf{B}$, therefore, to minimize (3.2.1), trace$(\boldsymbol{X}_2^T\boldsymbol{X}_1\boldsymbol{B})$ must be maximised.

Expressing $\boldsymbol{X}_2^T\boldsymbol{X}_1$ in terms of its singular value decomposition $\boldsymbol{X}_2^T\boldsymbol{X}_1 = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ gives

$$\text{trace}(\boldsymbol{X}_2^T\boldsymbol{X}_1\boldsymbol{B}) = \text{trace}(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{B}) = \text{trace}(\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{B}\boldsymbol{U}) = \text{trace}(\boldsymbol{\Sigma}\boldsymbol{H})$$

with $\boldsymbol{H} = \boldsymbol{V}^T\boldsymbol{B}\boldsymbol{U}$. As $\boldsymbol{H}$ is the product of orthogonal matrices it is orthogonal as well. Also, trace$(\boldsymbol{\Sigma}\boldsymbol{H})$ can be written as:

$$\text{trace}(\boldsymbol{\Sigma}\boldsymbol{H}) = \sum_{i=1}^{P} h_{ii}\sigma_i. \tag{3.2.3}$$

The singular values σ_i are non-negative, therefore (3.2.3) is maximum when $h_{ii} = 1$ (the maximum value attainable for the elements of an orthogonal matrix) for all i. Therefore $\boldsymbol{H} = \boldsymbol{I}$, giving $\boldsymbol{I} = \boldsymbol{V}^T\boldsymbol{B}\boldsymbol{U}$ and finally,

$$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{U}^T. \tag{3.2.4}$$

The residual sum of squares is given by

$$\|\boldsymbol{X}_1\boldsymbol{B} - \boldsymbol{X}_2\|^2 = \|\boldsymbol{X}_1\|^2 + \|\boldsymbol{X}_2\|^2 - 2\text{trace}(\boldsymbol{\Sigma}) \tag{3.2.5}$$

Expression (3.2.1) can be expanded with an isotropic scaling factor s to give $\|s\boldsymbol{X}_1\boldsymbol{B} - \boldsymbol{X}_2\|^2$, which can be expanded as:

$$\|s\boldsymbol{X}_1\boldsymbol{B} - \boldsymbol{X}_2\|^2 = s^2\|\boldsymbol{X}_1\|^2 + \|\boldsymbol{X}_2\|^2 - 2s \times \text{trace}(\boldsymbol{\Sigma}) \tag{3.2.6}$$

Table 3.2: Decomposition of the total sum of squares in orthogonal Procrustes analysis

Source of Variation	Sum of Squares
Fitted	$2s \times \text{trace}(\boldsymbol{\Sigma})$
Residual	$\|s\boldsymbol{X}_1\boldsymbol{B} - \boldsymbol{X}_2\|^2$
Total	$s^2\|\boldsymbol{X}_1\|^2 + \|\boldsymbol{X}_2\|^2$

Differentiating (3.2.6) with respect to s to find the optimal scaling factors for minimizing (3.2.6) leads to (Cox and Cox, 2001)

$$\hat{s} = \frac{\text{trace}(\boldsymbol{\Sigma})}{\|\boldsymbol{X}_1\|^2}. \tag{3.2.7}$$

It follows from (3.2.6) that the total sum of squares $s^2\|\boldsymbol{X}_1\|^2 + \|\boldsymbol{X}_2\|^2$ can be partitioned into a fitted and a residual component as shown in Table 3.2.

Orthogonal Procrustes Analysis can be generalized to K configurations (groups) optimally fitted to their group average configuration $\boldsymbol{G}$ defined as:

$$\boldsymbol{G} = \frac{1}{K}\sum_{k=1}^{K} s_k\boldsymbol{X}_k\boldsymbol{B}_k. \tag{3.2.8}$$

Transformation is performed by ensuring that all configurations have the same centroid, and this is taken as the origin $\boldsymbol{O}$. The quantity s_k denotes the isotropic scaling factor for the k-th configuration. Rotation or reflection of the k-th configuration about $\boldsymbol{O}$ is affected by the orthogonal matrix $\boldsymbol{B}_k$. Therefore, Generalized Orthogonal Procrustes Analysis (GOPA) minimizes

$$\sum_{k=1}^{K} \|s_k\boldsymbol{X}_k\boldsymbol{B}_k - \boldsymbol{G}\|^2. \tag{3.2.9}$$

To avoid the trivial solution of all $s_k = 0$ the usual equality constraint for the total size before and after scaling was adopted i.e.,

$$\sum_{k=1}^{K} \|\boldsymbol{X}_k\|^2 = \sum_{k=1}^{K} \|s_k\boldsymbol{X}_k\boldsymbol{B}_k\|^2. \tag{3.2.10}$$

The basic calculations needed for performing a GOPA as discussed in (Gower and Dijksterhuis, 2004, section 9.1.4), were implemented. Generally, the matrices $\boldsymbol{X}_k$ are centred and scaled to $\text{trace}(\boldsymbol{X}_k^T\boldsymbol{X}_k) = 1$ such that each column of $\boldsymbol{X}_k$ has the same sum of squares i.e., $1/p_k$ where p_k denotes the number of columns of $\boldsymbol{X}_k$. This is termed P_k scaling. In the current application, since the $\boldsymbol{X}_k$ matrices contained the same variables performing P_k-scaling was not required. The trains do however contain different number of gasifiers, and therefore padding was needed to ensure all matrices had the same number of

columns. If the maximum number of columns over all p_k columns is defined as $max(p_k)$ then padding is performed by adding $p_k - max(p_k)$ columns of zeros to each matrix $\boldsymbol{X}_k$.

3.2.4 Results

3.2.4.1 Selection of the reference set

GOPA was applied to the production facility outlined in Figure 3.1 for the selection of a reference set for multivariate process monitoring of all the production processes simultaneously. The data from the production facility consist of three different entities:

1. First, the groups; in this case the different production trains.
2. Secondly the variables captured for each train.
3. Lastly the time dependency.

In the selection of the reference set, the optimal train as well as the optimal combination of weeks will be selected.

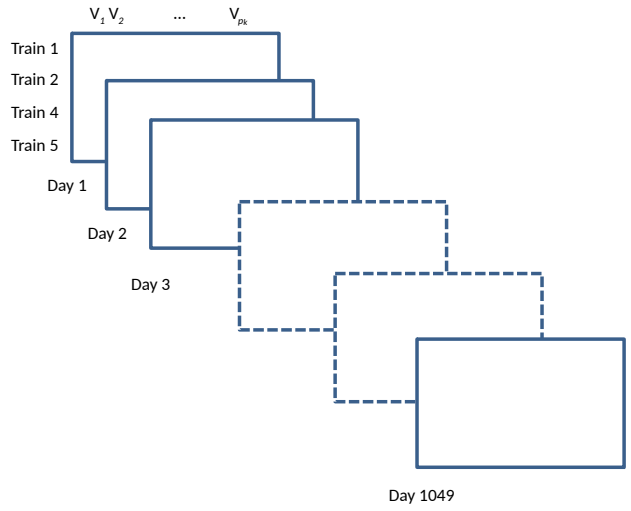


Figure 3.3: GOPA data structure

In applying the GOPA procedure outlined above it is important to formulate the problem correctly. Daily average data were collected for each production process. The days were specified as K different configurations, with $K = 1049$, and the trains were constructed as the n rows of the K matrices. Each train consists of either 10 or 11 reactors, and 11 variables are captured for each process. Thus, the data matrix is constructed with the process variables of all reactors on the trains as columns (here for example for a train

with 11 reactors $p_1 = p_2 = \cdots = p_{1049} = 11 \times 11 = 121$ columns) and the trains as rows for each time interval. In this example the units of time were days. Therefore, a data matrix is constructed for each day as summarized in Figure 3.3. The matrices $\boldsymbol{X}_k$ were centered and scaled to trace$(\boldsymbol{X}_k^T \boldsymbol{X}_k) = 1$ such that each column of $\boldsymbol{X}_k$ has the same sum of squares i.e., $1/p_k$ where p_k denotes the number of columns of $\boldsymbol{X}_k$ (either 110 or 121 depending on the number of reactors per train). Therefore, the group average configuration $\boldsymbol{G}$ is populated with the mean over trains and variables. Therefore, the dimension of $\boldsymbol{G}$ is $T \times max(p_k)$ where T is the number of trains (four in this study) and $max(p_k)$ is the maximum of the number of columns over all $\boldsymbol{X}_k$ (which will be 121 in this study). It will be demonstrated how the information from the GOPA output can be utilised to choose both the optimal train and the optimal combination of weeks for the reference set. Specifically, the aim is to select the group with the closest average distance to all the other groups for some distance measure. The group average configuration $\boldsymbol{G}$ is used to assess the closeness of each group to the group average configuration.

The data for day $k = 1, 2, \cdots, K$, viz $\boldsymbol{X}_k$, can also be written as $\boldsymbol{X}_{md}$ where $m = 1, 2, \cdots, M$ denotes the month number and $d = 1, 2, \cdots, D_m$ denotes the day number within month m. Therefore equation (3.2.9) can be rewritten as

$$\sum_{m=1}^{M} \sum_{d=1}^{D_m} \|s_{md} \boldsymbol{X}_{md} \boldsymbol{B}_{md} - \boldsymbol{G}\|^2 \tag{3.2.11}$$

where M is defined as the total number of months, and D_m is defined as the number of days in month m. The total GOPA sum of squares can be written as the sum of the squared contributions of the individual months. These monthly contributions provide insight into how well the individual months compare to the group average configuration $\boldsymbol{G}$. Due to plant upsets data were not available on all 1049 days for all the production processes. The sum of squares output of the GOPA criterion (3.2.11) consists of the sum of the sum of squares values for all $M \times D_m = K$ days. However, due to the uneven number of days in the months the mean of the daily contributions was calculated for each month m i.e.,

$$\frac{1}{D_m} \sum_{d=1}^{D_m} \|s_{md} \boldsymbol{X}_{md} \boldsymbol{B}_{md} - \boldsymbol{G}\|^2 \tag{3.2.12}$$

was calculated. The values from (3.2.12) are depicted in Figure 3.4 and Figure 3.5 for the Western and Eastern factory respectively. Note that due to confidentiality constraints the actual months were encoded as M01 to M35. The graphs are now used to select the optimal combination of months for the reference set by choosing the months with the smallest average GOPA sum of squares contributions. For the production facility under investigation it was decided to use one month of data for the reference set. As will be discussed in Section 3.3 the facility is monitored in real time with 15 minute averaged data.

The reference set will therefore consist of a month's record of 15 minute data for 11 variables on either 10 or 11 gasifiers. The data for one month should therefore be more than adequate for the reference set. The results for selecting combinations of one or two months of data are summarized in Table 3.3. These results indicate that if for instance two months of data are required as reference set for the western factory, M02 and M01 form the optimal combination. The optimal one month period is M02. The optimal two months are highlighted in red in Figure 3.4 and Figure 3.5.

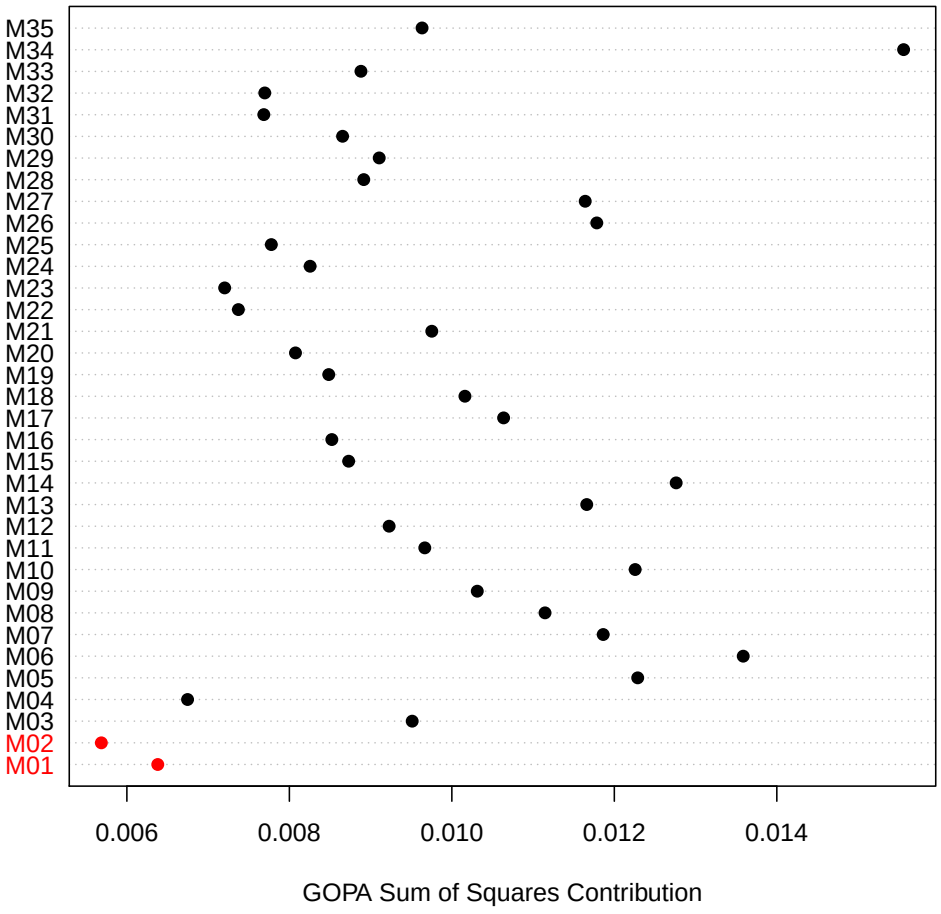


Figure 3.4: Average monthly GOPA sum of squares contribution for Western factory (the optimal two months are highlighted in red)

The relative sizes of the isotropic scaling factors s_k are valuable additional information provided by the GOPA analysis. In this study it was observed that the days underwent similar isotropic scaling. Figure 3.6 is provided as

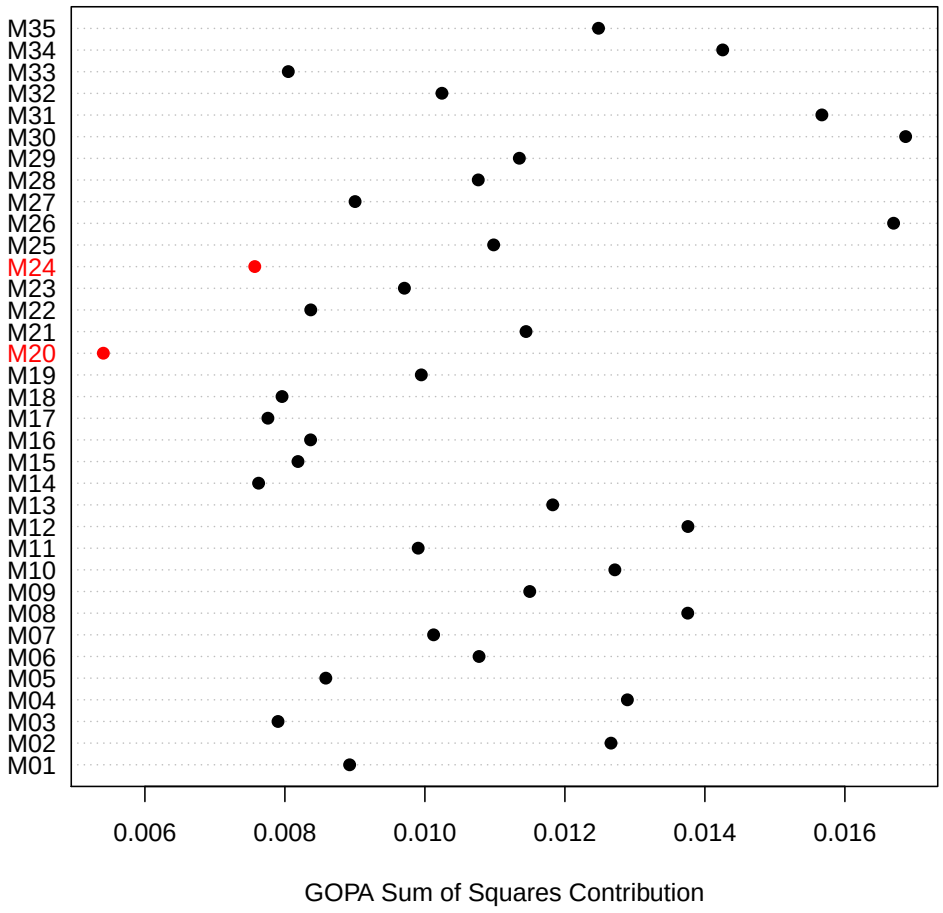


Figure 3.5: Average monthly GOPA sum of squares contribution for Eastern factory (the optimal two months are highlighted in red)

Table 3.3: Optimal combination of Months

Side	Month 1	Month 2
West	M02	M01
East	M20	M24

an example of a boxplot of the isotropic scaling factors for the optimal one and two month period (M20 and M24) for the Eastern factory. It is clear that the isotropic scaling factors are similar, and close to one. One outlying value can be observed for the optimal two month combination. However, this value is still close to 1 at 0.982. The specific day (25) corresponding to the outlier value also exhibited the highest GOPA sum of squares value.

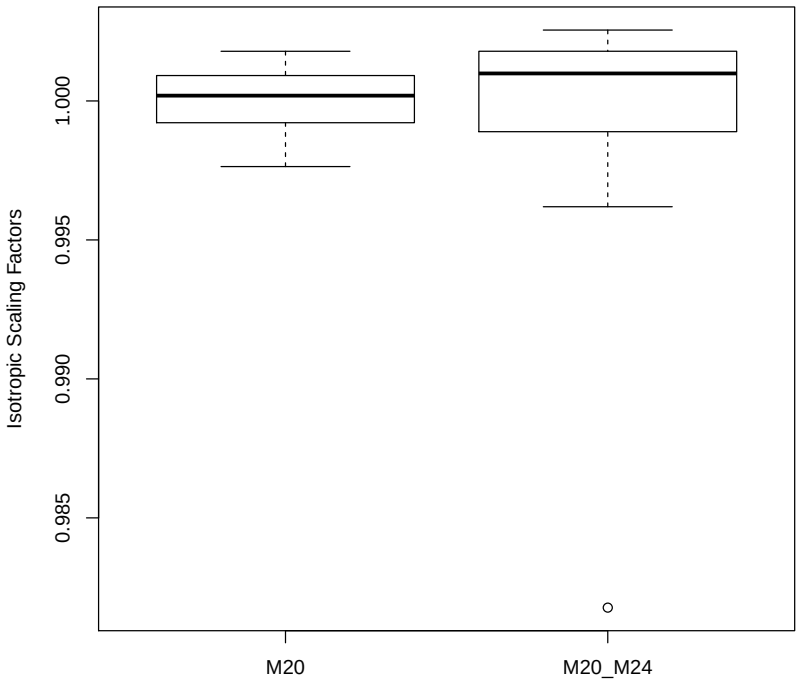


Figure 3.6: Isotropic scaling factors for optimal one and two month combinations for the Eastern factory

Table 3.4: Euclidean distance of trains from overall centroid $\boldsymbol{O}$.

Side	Months	TR1	TR2	TR4	TR5
West	1	0.5118	0.4826	0.5116	0.4885
West	2	0.5112	0.4854	0.5167	0.4809
East	1	0.4832	0.4909	0.5028	0.5184
East	2	0.4811	0.5089	0.4988	0.5050

Given that the optimal reference set has been determined for the months, the optimal train can be selected from the output of the GOPA analysis. Specifically, the group average configuration $\boldsymbol{G}$ obtained for the optimal GOPA configuration can be used to select the train closest to the overall centroid $\boldsymbol{O}$ of the optimal $\boldsymbol{G}$. The GOPA objective function (3.2.12) was minimized for each of the optimal one and two month combinations specified in Table 3.3. Expressing $\boldsymbol{G}$ in terms of its singular value decomposition $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, then according to the Eckart Young Theorem an optimal r^* dimensional representation of $\boldsymbol{G}$ can be obtained from the first r^* columns of $\boldsymbol{V}$ as $\boldsymbol{G}_{r^*} = \boldsymbol{G}\boldsymbol{V}_{r^*}$. For each $\boldsymbol{G}$ a two dimensional representation can thus be constructed by plotting

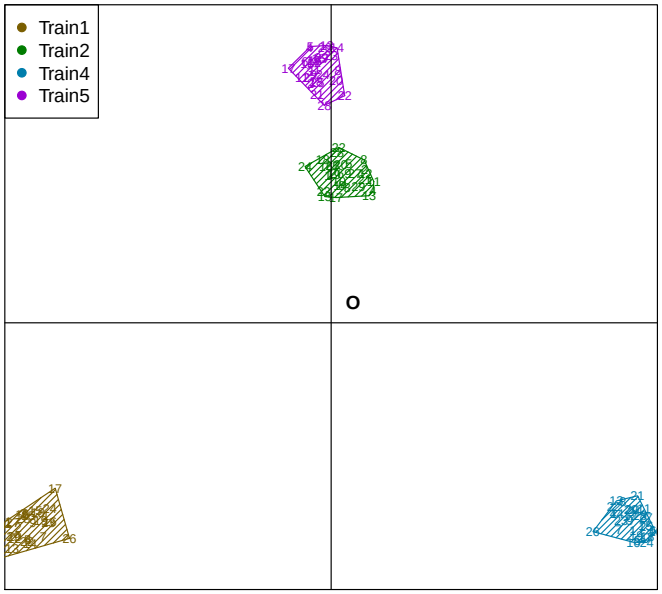
the first two principal components. A new sample $\boldsymbol{x}$ can be projected onto this lower dimensional subspace by utilising $\boldsymbol{x}_2 = \boldsymbol{x}\boldsymbol{V}_2$. Therefore, $\boldsymbol{G}_2 = \boldsymbol{G}\boldsymbol{V}_2$ was obtained for each of the optimal one and two month combinations from the respective group average $\boldsymbol{G}$ from the GOPA output. Additionally, the optimal configuration for each day $\boldsymbol{X}_k$ (Figure 3.3) from the GOPA output was projected on the same two dimensional plane by $\boldsymbol{X}_{kf2} = \boldsymbol{X}_{kf}\boldsymbol{V}_2$ where $\boldsymbol{X}_{kf} = s_k\boldsymbol{X}_k\boldsymbol{B}_k$. Principal component biplots will be discussed in detail in Section 3.3.

The results are provided in Figures 3.7a to 3.7d for the optimal month combinations for each side of the factory respectively. The convex hull, representing the data for the different days from the final GOPA results, is added to the plots to provide an indication of the variation over the days. For the Eastern factory it is clear from Figures 3.7c and 3.7d that train one is consistently the closest to the overall centroid $\boldsymbol{O}$ represented by the intersection of the two lines added to the plots. Therefore, train one is selected as the reference set for the multivariate monitoring of the multiple processes for all the trains on the eastern factory. Note that selecting train one as the reference set implies that all the production processes (reactors) on train one are used as the reference set. For the Western factory train two is closest to the overall centroid $\boldsymbol{O}$ for a one month period and train five is closest to the overall centroid for the two month period. Since a one month period is of interest, train two will be selected as the reference set for the multivariate monitoring of the multiple processes for all the trains on the Western factory.

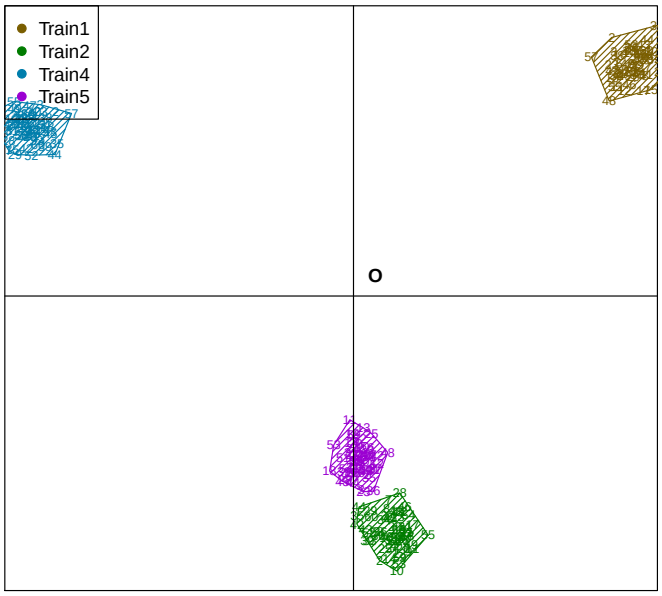
To conclude the analysis, the quality (variance explained/modeled) of each two dimensional approximation is summarized in Table 3.5. All the two dimensional approximations account for approximately 69% of the variation in the data, and it can therefore be concluded that the two dimensional representation approximates the full dimensional space satisfactorily. In addition the Euclidean distance for each Train to the overall centroid has been calculated for the full dimensional space. These distances are given in Table 3.4 and confirm the results from the two dimensional visualization, as the relative Euclidean distances correspond closely to those shown in Figures 3.7a to 3.7d. See Arnold *et al.* (2007) for illustrations of various enhancements to the graphical output of GOPA.

Table 3.5: Biplot quality of the two-dimensional PCA plot for each side and month combinations (Figures 3.7a - 3.7d)

Side	One Month	Two Months
West	69.18%	69.50%
East	69.06%	70.46%

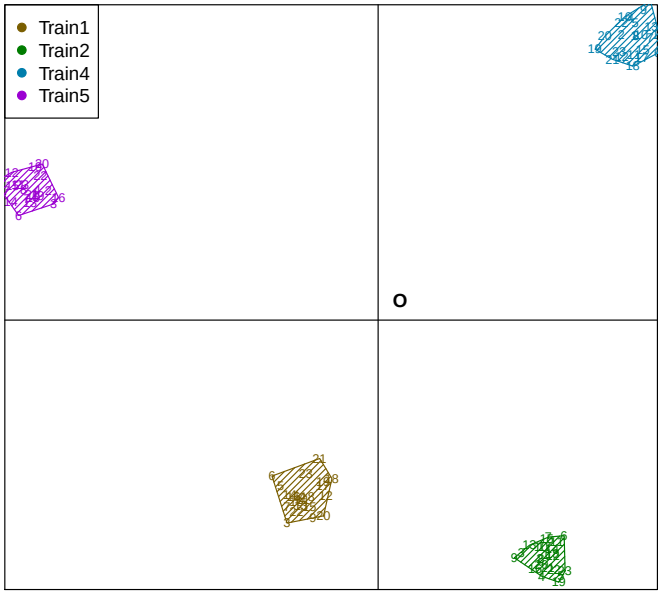


(a) Optimal one month for Western factory

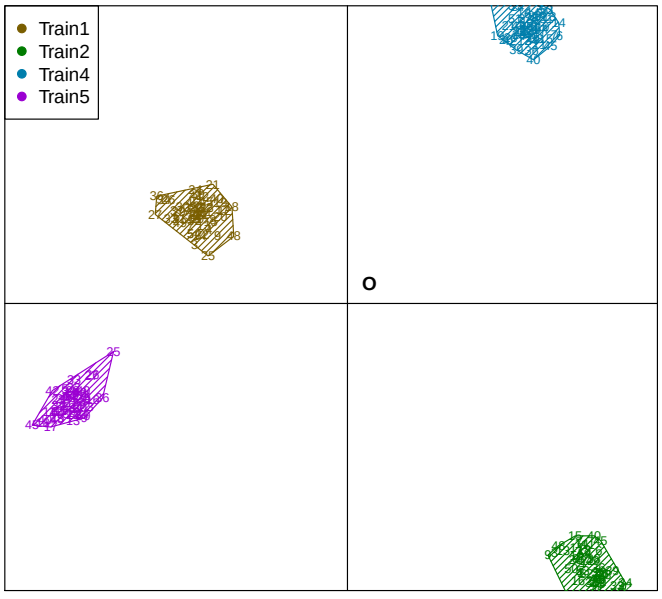


(b) Optimal two months for Western factory

Figure 3.7: PCA plots of group averages including the variation for each train for the optimal one and two month combinations (the overall centroid ***O*** is depicted at the intersection of the two lines on each plot)



(c) Optimal one month for Eastern factory



(d) Optimal two months for Eastern factory

Figure 3.7: PCA plots of group averages including the variation for each train for the optimal one and two month combinations (the overall centroid ***O*** is depicted at the intersection of the two lines on each plot)

3.2.5 Interpretation

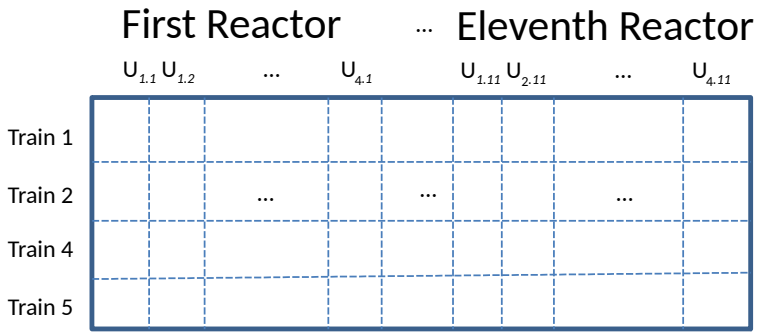


Figure 3.8: Data structure for calculating the group average matrix, $\boldsymbol{G}$. Variables named according to Table 3.1

Biplot axes representing the original measured variables can be added to the PCA plots to evaluate the effect of the variables on the differences between the groups (Gower *et al.*, 2011). Each train consists of 10 or 11 production processes, and 11 variables are captured for each process. Due to the methodology followed for the GOPA analysis, eleven sets of eleven axes can therefore be added to the PCA plot i.e., set one for the first process, and set two for the second process on the train (see Figure 3.8). This is illustrated in Figures 3.9 and 3.10 where the PCA biplot axes are added to the plot for the optimal month. The labels for the axes are written on the side where the variable is at its maximum. Note the variable types are provided in Table 3.1. Due to the large number of axes it is a lot more cluttered than the small scale example in Coetzer *et al.* (2014). To aid in the interpretation of the plots only the axes with axis predictivity values above 80% have been added to the plots (see Section 3.3.2 for a detailed discussion of PCA axis predictivity). Some subtle trends can however still be observed in these plots.

Specifically, from Figure 3.9 it can be observed that variables S7 and U4 are generally clustered on the left hand side of the plot. This could be an indication that these variables are generally higher for train five. This is an interesting observation from a process perspective as these two variables are indicative of the load on the reactors, and it can therefore be deduced that train five is operated with higher load than the remaining three trains. In addition, there is a slight clustering of the S5 variable toward the top right of the plot. This is a stability variable where lower values are desired. It would therefore appear that train four is operating more unstable compared to the other trains. This finding can be confirmed by noting that train four will project low on the S7 and U4 variables previously discussed, and will therefore run at lower loads. Some clustering of the S6 variable appears on the bottom

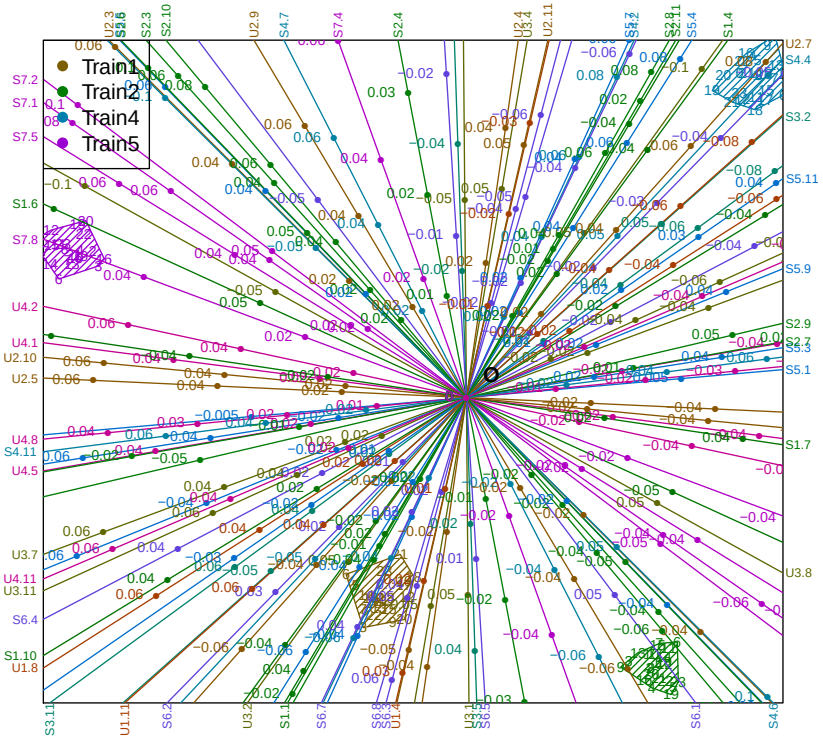


Figure 3.9: PCA biplot of group averages with variable axes for the optimal month for the Eastern factory

left of the graph. This is a stability variable that is linked to high loads, and it is therefore expected that gasifiers running at higher loads will experience higher levels of the S6 variable.

Similarly from Figure 3.10 some clustering of the U3 variable can be observed on the left side of the plot. U3 is a cooling agent and this could be an indication that train one is running at higher temperatures compared to the other trains. The S2 observations (a temperature variable) are clustered at the top of the plot. The S6 measurements are largely clustered on the right side of the plot. Careful consideration of this graph will yield more information.

From the above discussion it is clear that GOPA together with the appropriate graphical representations in the form of biplots provide important information and quantification of the relationships between the variables on the production facility. Our results obtained thus support our recommendation for the GOPA procedure to facilitate insight into the whole monitoring process and could therefore be considered to be of some consequence in the multivariate process monitoring literature.

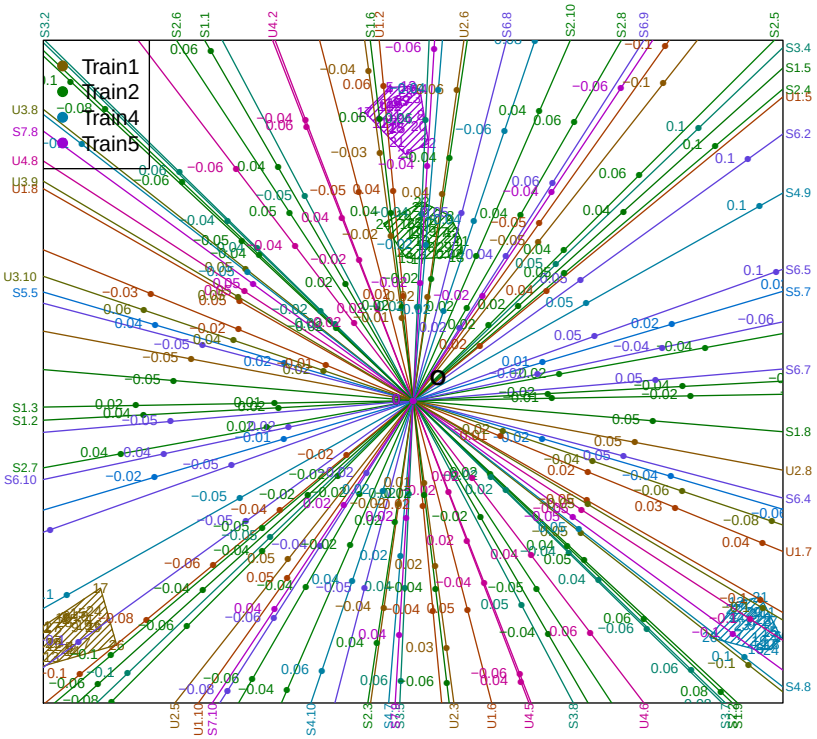


Figure 3.10: PCA biplot of group averages with variable axes for the optimal month for the Western factory

3.2.6 Conclusions for Reference Set Selection

In this section a new methodology was proposed and illustrated for the selection of an optimal reference set for real time multivariate monitoring of many mechanically identical production processes across different production trains for a very complex coal gasification production facility. Specifically, the production facility studied consists of 84 production processes grouped into eight production trains of 10 or 11 processes each split into two identical sites (Figure 3.1). Data on 11 process variables were captured in real time and used in the multivariate monitoring of the production processes. The aim was to select one production train for a specified combination of a number of months as the reference set for expected performance for each site.

GOPA was applied as a criterion for the selection of the optimal train and the optimal combination of the number of weeks as the reference set for all the production processes. PCA analyses and biplot displays were used to visualize and to interpret the results from the GOPA analysis. These interpretations provide important insight into and quantification of the relationships between the variables on the production facility. The way GOPA has been applied for reference set selection, and the accompanying PCA analyses is a novel approach to multivariate process monitoring.

The steps for determining the optimal reference set for multivariate monitoring of multiple production processes are summarized in Figure 3.11.

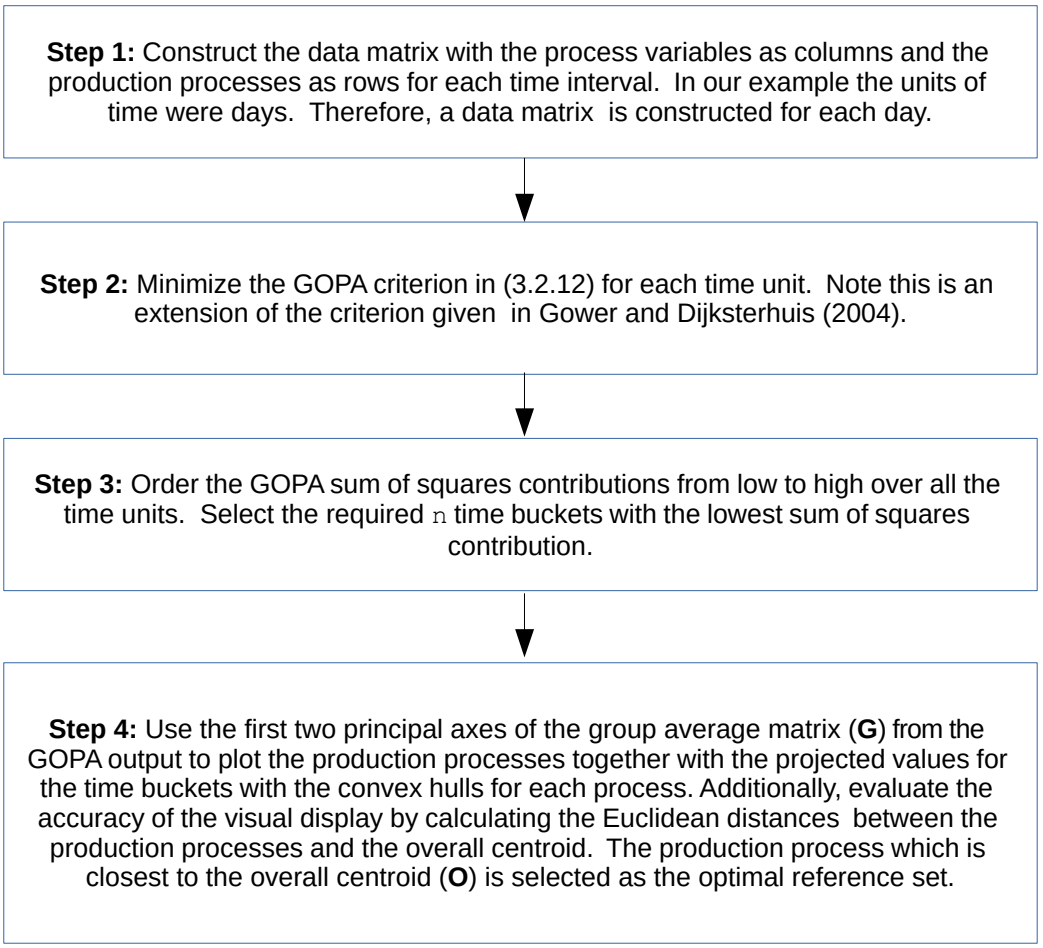


Figure 3.11: Steps for determining the optimal reference set for multivariate monitoring of multiple parallel processes

3.3 Multivariate process monitoring using the PCA biplot

Given the optimal reference set for process monitoring of the multiple parallel production processes, the PCA biplot methodology as described by Gower *et al.* (2011) is applied to the multivariate monitoring of each production process. Aldrich *et al.* (2004) and Sparks *et al.* (1997) discussed numerous advantageous of biplots for process monitoring. The biplot provides the ability to detect deviations from expected performance, as well as which variables are responsible for the deviation. Therefore, the implementation of monitoring biplots in real time provides the engineer with information on process variables for bringing the process back to expected or target performance. However, in this study, the application of monitoring biplots has been extended to multiple parallel production processes.

To demonstrate the implementation of the selected reference set for multivariate monitoring of multiple production processes, the optimal month was selected (Section 3.2) as the reference set for the monitoring biplots. In this section only the process monitoring of the Eastern factory is considered as the implementation will be similar for the Western factory. All the production processes on all the trains are monitored in real time using the M20 data from train one (i.e., all processes on train one) as the reference set for expected performance of the processes. The real time monitoring is currently performed on 15 minute aggregate data, and therefore the reference set will consist of 15 minute aggregate data for the ten production processes (gasifiers) on train one for all the days in M20. The data were filtered according to specifications provided by the production engineer (the filtering and ranges of the data will be discussed in more detail in Chapter 4).

In the next section the underlying algebra of the PCA biplot as well as some measures of the predictive power of the plots will be briefly reviewed.

3.3.1 PCA Biplot

Before presenting the underlying algebra for the PCA biplot, some necessary notation will be specified. Let $\boldsymbol{U}$ be an $n \times n$ orthogonal matrix with $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$. Therefore, the sum of squares of each column of $\boldsymbol{U}$ is equal to 1, and the columns of $\boldsymbol{U}$ are orthogonal (or perpendicular) to one another i.e., their scalar products are 0. Similarly, the sum of squares of each row of $\boldsymbol{U}$ is equal to 1, and the rows of $\boldsymbol{U}$ are orthogonal to one another. A rectangular $n \times p$ matrix $\boldsymbol{U}$ with $p < n$ is orthonormal if $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_p$, but $\boldsymbol{U}\boldsymbol{U}^T \neq \boldsymbol{I}_n$. Therefore, the sum of squares of each of the columns of an orthonormal matrix $\boldsymbol{U}$ is equal to 1, and its columns are orthogonal to one another.

The singular value decomposition (SVD) plays a central role in multivariate statistics (Gower *et al.*, 2011; Gower and Hand, 1996; Greenacre, 2010;

Greenacre and Primicerio, 2014). The SVD of a $n \times p$ matrix $\boldsymbol{X}$ with $n \geq p$ is defined as

$$\boldsymbol{X}_{n\times p} = \boldsymbol{U}^*_{n\times n}\boldsymbol{\Sigma}^*_{n\times p}(\boldsymbol{V}^*)^T_{p\times p} \tag{3.3.1}$$

where $\boldsymbol{U}^*$ is a $n \times n$ orthogonal matrix with columns known as the left singular values of $\boldsymbol{X}$, and $\boldsymbol{V}^*$ is a $p \times p$ orthogonal matrix with columns known as the right singular values of $\boldsymbol{X}$. The matrix $\boldsymbol{\Sigma}^*$ is of the form

$$\boldsymbol{\Sigma}^*_{n\times p} = \begin{bmatrix} \boldsymbol{\Sigma}_{r\times r} & \mathbf{0}_{r\times(p-r)} \\ \mathbf{0}_{(n-r)\times p} & \mathbf{0}_{(n-r)\times(p-r)} \end{bmatrix} \tag{3.3.2}$$

where r denotes the rank of $\boldsymbol{X}$, and an $r \times r$ diagonal matrix with the diagonal elements the non-zero singular values of $\boldsymbol{X}$. The singular values are non-negative, and specified in decreasing order i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$. Therefore, (3.3.1) can be written as

$$\boldsymbol{X} = \boldsymbol{U}_{n\times r}\boldsymbol{\Sigma}_{r\times r}\boldsymbol{V}^T_{r\times p} \tag{3.3.3}$$

where $\boldsymbol{U}$ is a $n \times r$ orthonormal matrix consisting of the first r columns of $\boldsymbol{U}^*$ and $\boldsymbol{V}$ is a $p \times r$ orthonormal matrix consisting of the first r columns of $\boldsymbol{V}^*$. In this study, all references to SVD refer to the reduced form in (3.3.3).

The non-zero singular values of $\boldsymbol{X}$ are the positive square roots of the non-zero eigenvalues of the eigenvalue (spectral) decomposition of the $p \times p$ square symmetric matrix $\boldsymbol{X}^T\boldsymbol{X}$. Therefore,

$$\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T \tag{3.3.4}$$

where $\boldsymbol{V}$ is identical to $\boldsymbol{V}$ in (3.3.3).

In this study the optimal approximation of an $n \times p$ data matrix $\boldsymbol{X}$ of rank r in fewer (say r^*) dimensions is of interest. Let $\hat{\boldsymbol{X}}_{r^*}$ of rank r^* be the approximation of $\boldsymbol{X}$ in r^* ($r^* < r$) dimensions. Then, Gower *et al.* (2011) calculated $\hat{\boldsymbol{X}}_{r^*}$ by the minimisation of

$$\text{trace}[(\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*})^T(\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*})] = \|(\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*})\|^2. \tag{3.3.5}$$

The solution to this least-squares problem is given by the Eckart-Young theorem (Eckart and Young, 1936). According to this theorem, the least squares problem is minimised in r^* dimensions by the r^* dimensional approximation for $\boldsymbol{X}$

$$\hat{\boldsymbol{X}}_{r^*} = \boldsymbol{U}_{r^*}\boldsymbol{\Sigma}_{r^*}\boldsymbol{V}^T_{r^*} \tag{3.3.6}$$

where $\boldsymbol{U}_{r^*}$ is a $n \times r^*$ orthonormal matrix with r^* the first r^* columns of $\boldsymbol{U}$, $\boldsymbol{V}_{r^*}$ is a $p \times r^*$ orthonormal matrix (for $r_* < p$) with r^* the first r^* columns of $\boldsymbol{V}$, and $\boldsymbol{\Sigma}_{r^*}$ is a $r^* \times r^*$ diagonal matrix.

The coordinates of the r^* dimensional approximation of $\boldsymbol{X}$ are given by $\boldsymbol{U}_{r^*}\boldsymbol{\Sigma}_{r^*} = \boldsymbol{X}\boldsymbol{V}_{r^*}$. In the PCA literature $\boldsymbol{X}\boldsymbol{V}_{r*}$ is known as the scores of the

PCA. The directions of the PCA biplot axes are given by the rows of $\boldsymbol{V}_{r^*}$, also known as the loadings.

Gower and Hand (1996) extended the biplot methodology to include calibration of the axes. Calibrations are added to the axes as follows (Alves, 2012; Gower *et al.*, 2011; Gower and Hand, 1996): Given a vector $\boldsymbol{\mu}_i$ of k scale values for the ith biplot axis and a vector $\boldsymbol{e}_i$ of length p containing 0's except for a 1 as the ith value, the coordinates of the scale values in the r^* dimensional representation of $\boldsymbol{X}$ can be obtained by

$$\frac{\mu_{ij}\boldsymbol{e}_i^T\boldsymbol{V}_{r^*}}{\boldsymbol{e}_i^T\boldsymbol{V}_{r^*}\boldsymbol{V}_{r^*}^T\boldsymbol{e}_i} \tag{3.3.7}$$

for $i = 1, \cdots, p$ and $j = 1, \cdots, k$.

Biplots are discussed in detail in Greenacre (2010), Gower and Hand (1996) and Gower *et al.* (2011).

3.3.2 PCA Biplot Measures of fit

According to Gower *et al.* (2011) the overall quality of the PCA biplot display is defined as:

$$\frac{\sum_{i=1}^{r^*}\sigma_i^2}{\sum_{i=1}^{p}\sigma_i^2}. \tag{3.3.8}$$

However, the overall quality of the biplot display does not contain any information about the quality of the representation of the variables in r^* dimensions.

The projection of unit points $\boldsymbol{I}$ on the coordinate axes can be written as $\boldsymbol{I}\boldsymbol{V}_{r^*}$, which gives p points that may be regarded as representing the variables. $\boldsymbol{V}$ is orthogonal, and therefore its rows have unit sums of squares. Thus, the sums of squares of the rows of $\boldsymbol{V}_{r^*}$ measures the adequacy of the representation for each variable (Gower *et al.*, 2011). Therefore, the adequacy for the ith variable is given by

$$\text{adequacy} = \sum_{j=1}^{r^*} v_{ij}^2 \tag{3.3.9}$$

which is equal to the ith diagonal element of $\text{diag}(\boldsymbol{V}_{r^*}\boldsymbol{V}_{r^*}^T)$. The maximum adequacy is equal to 1 when $r^* = p$. Adequacy measures how closely the endpoint of the ith vector $\boldsymbol{v}_i$ lies to the circumference of the unit circle in the r^* dimensional plane of approximation (Gardner-Lubbe *et al.*, 2008).

Gardner-Lubbe *et al.* (2008) argued that the adequacy, although popular, has limitations. An example is presented where the sample points lie on a plane in a three dimensional space. Therefore, the samples will be represented perfectly in a two dimensional approximation, whereas the three variables can not be represented perfectly in two dimensions. They stated that it is generally of more interest to know how close the points represented by the rows of $\boldsymbol{X}$ are to an approximating plane than how close the axes are.

As an alternative to the adequacy measure Gardner-Lubbe *et al.* (2008) proposed the axis predictivity, which measures the degree to which the columns of the approximation $\hat{\boldsymbol{X}}_{r^*}$ agree with the columns of the exact $\boldsymbol{X}$. The name axis predictivity is derived from the fact that it minimises the squared differences between the actual measurements of the variable and the predicted values of that variable as read off from the biplot axis. The axis predictivities are expressed as the ratio of the sums of squares $(\hat{\boldsymbol{X}}_{r^*}^T\hat{\boldsymbol{X}}_{r^*})$ to the total sums of squares $(\boldsymbol{X}^T\boldsymbol{X})$. Therefore, axis predictivities are defined as the diagonal elements of the matrix $\boldsymbol{\Pi}_{p\times p}$ given by

$$\boldsymbol{\Pi} = \text{diag}(\hat{\boldsymbol{X}}_{r^*}^T\hat{\boldsymbol{X}}_{r^*})[\text{diag}(\boldsymbol{X}^T\boldsymbol{X})]^{-1} \tag{3.3.10}$$

$$= \text{diag}(\boldsymbol{V}_{r^*}\boldsymbol{\Sigma}_{r^*}^2\boldsymbol{V}_{r^*}^T)[\text{diag}(\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T)]^{-1} \tag{3.3.11}$$

If $\boldsymbol{X}$ is standardised then (3.3.10) and (3.3.11) simplify to give

$$\boldsymbol{\Pi} = \text{diag}(\hat{\boldsymbol{X}}_{r^*}^T\hat{\boldsymbol{X}}_{r^*}) \tag{3.3.12}$$

$$= \text{diag}(\boldsymbol{V}_{r^*}\boldsymbol{\Sigma}_{r^*}^2\boldsymbol{V}_{r^*}^T) \tag{3.3.13}$$

As the axis predictivity values are expressed in terms of the fitted sums of squares over the total sums of squares, implicit reference is made to a residual sum of squares. Note, without the orthogonal decomposition the total sum of squares would not be equal to the sum of the fitted sum of squares and the residual sum of squares. Gardner-Lubbe *et al.* (2008) provided algebraic results to prove what they term Type B orthogonality:

$$\boldsymbol{X}^T\boldsymbol{X} = \hat{\boldsymbol{X}}_{r^*}^T\hat{\boldsymbol{X}}_{r^*} + (\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*})^T(\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*}) \tag{3.3.14}$$

Type B orthogonality shows that increasing the fitted sum of squares of a variable (the axis predictivity of the variable) reduces the residual sum of squares of the variable. Specifically, for standardised $\boldsymbol{X}$ the axis predictivity is

$$\boldsymbol{\Pi} = \text{diag}(\hat{\boldsymbol{X}}_{r^*}^T\hat{\boldsymbol{X}}_{r^*}) \tag{3.3.15}$$

$$= \boldsymbol{I} - \text{diag}((\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*})^T(\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*})) \tag{3.3.16}$$

Alves (2012) proposed the mean standard predictive error (mspe) criterion as a measure of predictive power of the biplot axes, given by:

$$\text{mspe} = \frac{\mathbf{1}^T\left|\boldsymbol{x}_{(1)} - \hat{\boldsymbol{x}}_{(1)}, \boldsymbol{x}_{(2)} - \hat{\boldsymbol{x}}_{(2)}, \cdots, \boldsymbol{x}_{(p)} - \hat{\boldsymbol{x}}_{(p)}\right|}{n} \tag{3.3.17}$$

where $\mathbf{1}^T$ is a $1 \times n$ vector of 1's, and $\boldsymbol{x}_{(i)}$ is the ith column of $\boldsymbol{X}$ and similarly $\hat{\boldsymbol{x}}_{(i)}$ is the ith column of $\hat{\boldsymbol{X}}_{r^*}$. The mspe criterion assumes the initial $\boldsymbol{X}$ was standardized to mean 0 and unit variance for each column. The mspe value for each axis (or variable) is the mean absolute residual value for the actual

against fitted values for each variable in $\boldsymbol{X}$. Therefore, a low mspe value represents good predictive power of the axis. Given that $\boldsymbol{X}$ is assumed to be standardised to mean 0 and unit variance the maximum mspe value will be equal to 1.

Comparing the axis predictivity values with the mspe values the following observations can be made:

- The mspe value for each axis (or variable) is the mean absolute residual value for the actual against fitted values for each variable in $\boldsymbol{X}$.
- The axis predictivity value for a variable for a standardised $\boldsymbol{X}$ is the fitted sum of squares value, or alternatively $\boldsymbol{I} - \text{diag}((\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*})^T(\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*}))$.

Therefore, both the mspe and the axis predictivity are based on the residual values $\boldsymbol{X} - \hat{\boldsymbol{X}}_{r^*}$, and it is expected that comparable results will be obtained by applying these two criteria to PCA biplots.

3.3.3 PCA Biplots for Monitoring

The PCA biplot can be extended to function as a multivariate monitoring graphic (Aldrich *et al.*, 2004; Coetzer *et al.*, 2014; Sparks *et al.*, 1997). If it can be assumed that the rows of the $n \times p$ reference set ($\mathbf{X}$) are approximately multivariate normally distributed with mean equal to $\bar{\mathbf{x}}$ and covariance matrix $\mathbf{S}$, it follows that the multivariate normal density is constant on surfaces where the Mahalanobis distance $(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{S}^{-1} (\mathbf{x} - \bar{\mathbf{x}})$ (with $\mathbf{x}$ a row of $\mathbf{X}$) is constant. That is, the rows of $\mathbf{X}$ lie on the surface of an ellipsoid centered at $\bar{\mathbf{x}}$.

It can then be shown that

$$(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{S}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) \sim \chi^2_p, \tag{3.3.18}$$

where χ^2_p denotes the chi-square distribution with p degrees of freedom, which is closely related to the T^2-statistic discussed in detail in Section 4.2.2. According to Gower *et al.* (2011) the $(1 - \alpha)\,\%$ concentration ellipse is defined by all observations that satisfy

$$(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{S}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) \leq \chi^2 (\alpha)\,. \tag{3.3.19}$$

Any observation that does not satisfy the above inequality is flagged for further inspection.

3.3.4 Results

3.3.4.1 Number of principal components to include

Alves (2012) discussed various strategies to determine the number of principal components. Specifically,

- Exclude the components for which the eigenvalues of the correlation matrix are less than one. This method was first proposed by Kaiser (1958), and is a special case of the more general methodology to exclude all the eigenvalues that are less than the average of all the eigenvalues. i.e., if λ_i is defined as the ith eigenvalue of $\boldsymbol{S}$ the average eigenvalue will be $\sum_i^p \frac{\lambda_i}{p}$. Note that the $\sum_i^p \lambda_i = \text{trace}(\boldsymbol{S})$ and therefore the average of the eigenvalues is equal to the average variance (Everitt and Hothorn, 2011).

- Retain the number of components that correspond to some percentage of the total information (typically 80%).

- Choose the number of components based on the scree test. The scree test entails plotting the eigenvalues λ_i against i and visually identifying the point where the slope change.

Greenacre and Primicerio (2014) page 223-224 discussed a formal methodology to specify the number of significant principal components using a permutation test. A brief outline of their methodology is provided here. In PCA the structure in the data is dependent on the correlation of the variables i.e., if there is no correlation between the variables there will be no structure in the data for PCA to capture. To test the null hypothesis that there is no correlation between the variables random data sets are generated by permuting the values in the columns of the data set. As an example, one random $n \times p$ data set (say $\boldsymbol{Z}$) is generated from the reference set $\boldsymbol{X}$ by sampling without replacement (permute) the data in column $\boldsymbol{X}_i$ to populate column $\boldsymbol{Z}_i$ for $i = 1, \cdots, p$. Two criteria i.e., percentage variance explained and eigenvalues of the correlation matrix, are then calculated on $\boldsymbol{Z}$, and the process is repeated j times. This will generate j results. The significance (p values) for each dimension is now calculated by taking the number of the generated criterion that is higher than the original criterion for each dimension. For example, if $j = 9999$ and one value is larger than the original criterion for dimension one the p value equals 0.0001. The p-value thus obtained is termed the achieved significance level (ASL).

For the reference set the eigenvalues of the correlation matrix were calculated. The results are shown in Figure 3.12. According to criterion 1, four principal components should be retained. The cumulative percentage variance explained was also calculated (Figure 3.13), and according to this criterion five principal components should be retained for more than 80% variability explained.

The permutation test was performed for $j = 9999$ for both the eigenvalue and the cumulative percentage variance explained, and equivalent results were obtained. All the values of the permuted data sets were smaller than those for the original data set up to and including four dimensions. For five and higher dimensions all the criterion values of the permuted data set were higher than the original data set. Box plots for both the eigenvalues of the correlation

matrix and the percentage variance explained are provided in Figures 3.14 and 3.15 respectively. The permuted values are depicted in the box plot, and the original values are highlighted in red. It was therefore concluded that four principal components should be retained for specifying the reference data set.

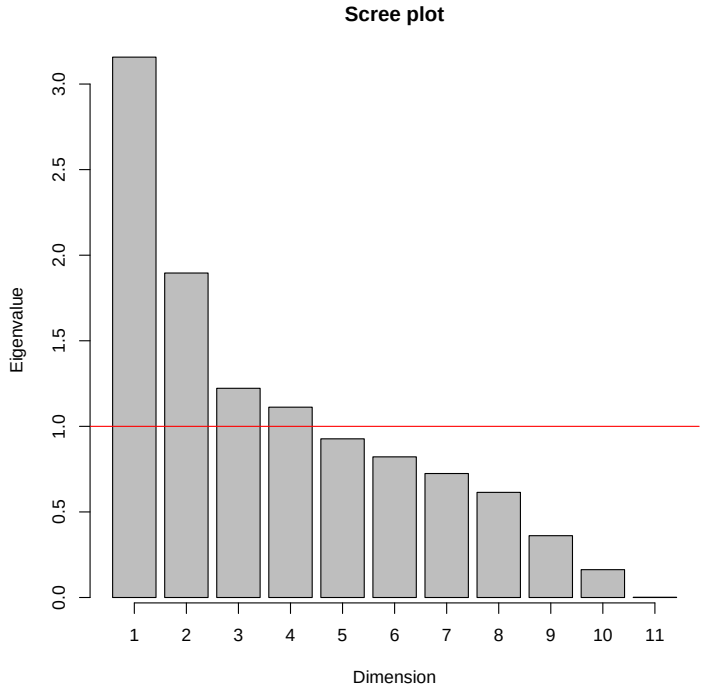


Figure 3.12: Scree Plot

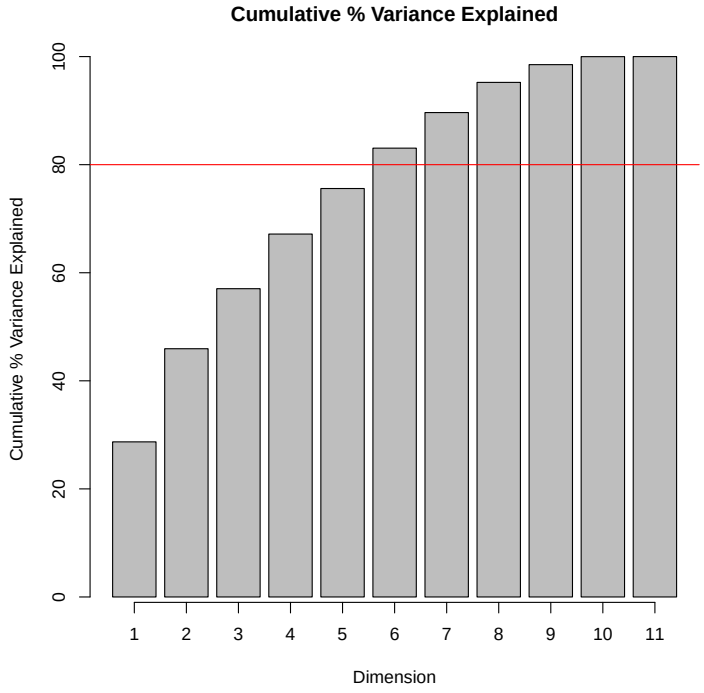


Figure 3.13: Cumulative percentage variance explained

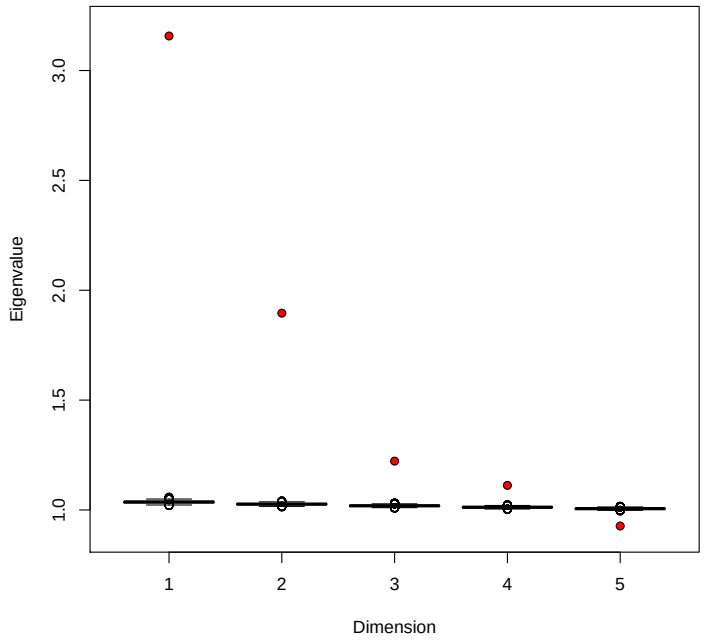


Figure 3.14: Permutation versus original eigenvalues (The permuted values are depicted in the box plot, and the original values are highlighted in red)

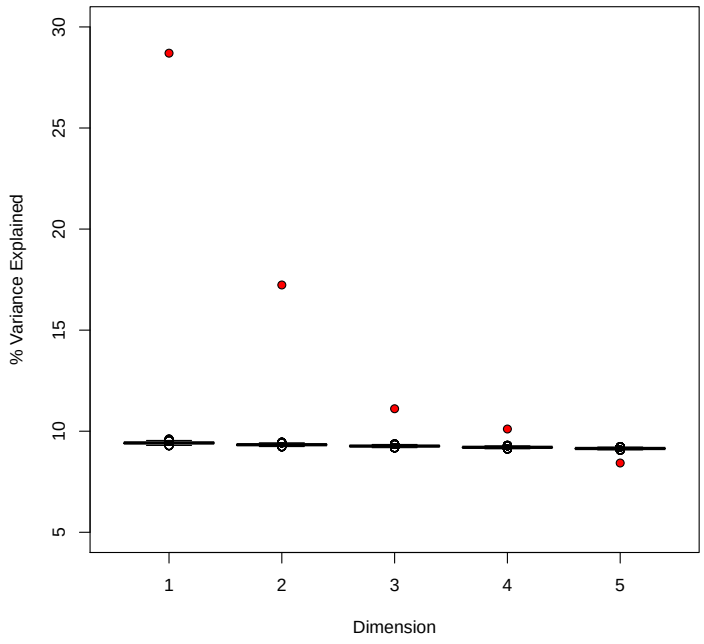


Figure 3.15: Permutation versus original variance explained (The permuted values are depicted in the box plot, and the original values are highlighted in red)

3.3.4.2 Axis predictivity and interpretation

Figures 3.16a to 3.16f depict the PCA biplots of the reference set of the Eastern factory as discussed in Section 3.2.4.1. Note that for all the biplots generated in this section the smaller numbered principal component will be represented on the x axis and the higher numbered principal component on the y axis. For example, in Figure 3.16a PC1 is represented along the x axis and PC2 along the y axis. The PCA biplot qualities are provided in Table 3.6 for the different combinations. One observation from the figures is that there seems to be two groups in the data and some additional scatter on the data along the first principal component. One possibility is that these points are outliers which should be screened from the reference set. However, there is information in this scatter. The data projects lower on variables U1, U4, and S7 which indicates low load conditions on the specific process unit for the reference set. From a process perspective these data are therefore expected and valid. This is one example where a purely data driven approach without taking into consideration the actual process health may lead to the possible exclusion of valid data. In Chapter 4 process versus data driven approaches will be discussed in detail. For the current application these data will be retained.

Although it is possible to interpret the biplots with all the axes included it will be demonstrated in the remainder of this section that:

- Using a predictivity measure to limit the axes on each plot guides the user in interpreting the data.
- The predictivity measures contain valuable information.
- Adding additional principal component combinations instead of only using the traditional plot of first and second principal components yields additional value to the interpretation of the data.

Both the axis predictivity (3.3.10) from Gardner-Lubbe *et al.* (2008) and the MSPE criteria (3.3.17) from Alves (2012) were implemented. The results are summarized in Table 3.7 and Table 3.8. For a first approximation a criterion of axis predictivity above 0.5 and MSPE below 0.5 were implemented. The results are highlighted in Table 3.7 and Table 3.8. The plots for the different principal component combinations are provided in Figure 3.17 and 3.18 respectively.

First, the results from Tables 3.7 and 3.8 are compared. It is clear that although the axis predictivity and MSPE results mostly agree there are some noticeable differences. This is most obvious for variables U1, S7 and U4 where for principal component combinations 2×3, 2×4 and 3×4 the axis predictivities for U1, S7 and U4 are close to zero, while the mspe values are still in the acceptance range. The remaining differences are mostly on borderline cases, for example U2 for principal component combination 1×2 is accepted by the predictivity criterion, but rejected by the mspe criterion. The mspe value is

Table 3.6: PCA biplot quality for Figures 3.16a to 3.16f

	PC (x-axis)	PC (y-axis)	Quality (%)
Figure 3.16a	1	2	45.94
Figure 3.16b	1	3	39.81
Figure 3.16c	1	4	38.81
Figure 3.16d	2	3	28.35
Figure 3.16e	2	4	27.34
Figure 3.16f	3	4	21.22

however 0.54 which is very close to the (arbitrary) cutoff value of 0.5. To aid in the interpretation of the results the PCA loadings for the first four principal components (the first four columns of $\boldsymbol{V}$) are provided in Table 3.9. Loadings with an absolute value larger than 0.4 are highlighted. As the data were mean centered and unit scaled the relative sizes of the loadings can be interpreted. In addition, the origin of the PCA biplot will be at $x = 0, y = 0$, and therefore positive loadings for a variable will lead to predicted values on the biplot axes increasing towards the edges of the top right quadrant of the biplot for the specific variable. Closer investigation of Tables 3.7 and 3.9 leads to the following observations:

1. The axis predictivities for variables U1, S7 and U4 are higher in all combinations where principal component one is present. This observation is confirmed by the loadings for principal component one, where all three these variables have relatively large loadings compared to the other variables. U1, S7 and U4 are variables that are all directly linked to reactor load. All the loadings are negative, and therefore in the same direction. Specifically, S7 is a control variable that is used to regulate the reactor load by directly influencing U4 and indirectly U1. The relationship between these variables will be explained in more detail in Chapter 4 Section 4.1.2. It can therefore be concluded that principal component one predominantly explains the variance in the reactor load variables.

2. The axis predictivities for U3, S1 and S3 are higher in all combinations where principal component two is present. This observation is confirmed by the loadings for principal component two. Variable U3 has a relatively large positive loading, and variables S1 and S3 have relatively large negative loadings. Therefore, it can be concluded that variable U3 is negatively correlated with variables S1 and S3. Note that in PCA the signs of the components are arbitrary, but the relative signs of the loadings on a component can be interpreted. Variables S1 and S3 are two temperature variables. S1 measures the temperature at the top of the reactor and S2 measures the temperature of the product leaving the reactor. Note that in the gasification process the reagent is introduced at the bottom of the reactor. Variable U3 measures the reactor cooling

agent level. A higher cooling agent level will lower the temperature of the reactor. It can therefore be concluded that principal component two predominantly represents relationship between the temperature at the top of the reactor and the amount of cooling agent applied.

3. The axis predictivities for U2 and S5 are higher in all combinations where principal component three is present. This observation is confirmed by the loadings for principal component three. Both variables U2 and S5 have relatively large negative loadings. Variables U2 regulates the ratio of variable U1 to variable U2 and a higher ratio will decrease the oxygen feed (U2) to the reactor. Variable S5 is a combustion side product that is linked to oxygen availability.

4. The axis predictivities for S2 and S4 are higher in all combinations where principal component four is present. This observation is confirmed by the loadings for principal component four. Both variables S2 and S4 have relatively large negative loadings. Variable S2 is a temperature variable in the lower part of the reactor and variable S4 is a variable related to the difficulty of the ash removal from the bottom of the reactor. Therefore, principal component four represents the temperature at the bottom of the reactor, with the accompanying changes in ash properties.

Even though some of the higher values are below the cutoff specification of 0.5 they are still significantly higher compared to the remaining values for the other variables. It is clear that the axis predictivity values are linked to the underlying structure in the data, and are correlated with the PCA loadings. In addition, the results could be validated from a process perspective. It can therefore be concluded that the structure in the principal components successfully captured the underlying dependencies of the gasification process. Therefore, the reference set selected is appropriate for monitoring the process. The axis predictivity values aided in highlighting the structure, and it is therefore suggested that predictivity is not only utilized to eliminate axes from the display, but also as an aid to analyze the underlying structure of the data. From the above analysis it was decided to utilize the axis predictivity values to choose the appropriate axes for the monitoring biplots. In addition, a cutoff value of 0.35 will be utilized to include all the relevant axes for each principal component combination in the study.

Table 3.7: Axis Predictivity

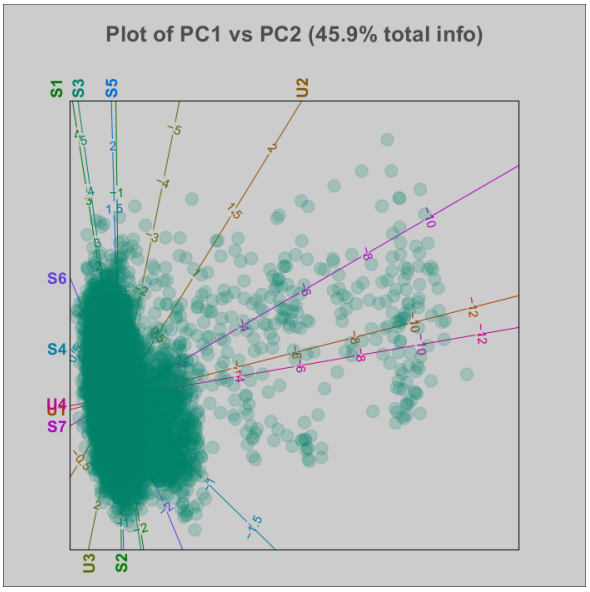
	PC1	PC2	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1	1	2	0.95	0.07	0.59	0.41	0.04	0.67	0.08	0.11	0.45	0.77	0.91
2	1	3	0.94	0.53	0.12	0.06	0.18	0.11	0.08	0.44	0.25	0.74	0.94
3	1	4	0.94	0.14	0.14	0.08	0.36	0.09	0.61	0.03	0.23	0.74	0.91
4	2	3	0.01	0.50	0.48	0.36	0.22	0.64	0.02	0.55	0.24	0.05	0.04
5	2	4	0.01	0.11	0.50	0.38	0.40	0.62	0.56	0.14	0.22	0.05	0.01
6	3	4	0.00	0.57	0.03	0.03	0.54	0.06	0.56	0.47	0.02	0.02	0.03

Table 3.8: Axis MSPE

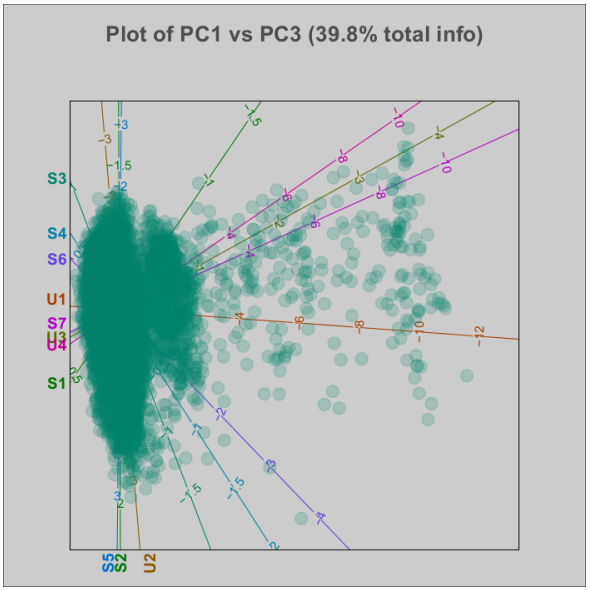
	PC1	PC2	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1	1	2	0.19	0.70	0.50	0.61	0.79	0.45	0.74	0.72	0.58	0.31	0.24
2	1	3	0.20	0.54	0.77	0.77	0.72	0.75	0.73	0.57	0.67	0.31	0.20
3	1	4	0.20	0.69	0.76	0.76	0.62	0.75	0.48	0.75	0.69	0.32	0.24
4	2	3	0.50	0.54	0.57	0.62	0.70	0.45	0.75	0.51	0.66	0.32	0.48
5	2	4	0.50	0.70	0.55	0.61	0.60	0.47	0.49	0.71	0.67	0.32	0.48
6	3	4	0.49	0.50	0.79	0.78	0.52	0.77	0.49	0.56	0.77	0.28	0.47

Table 3.9: PCA loadings for first four principal components

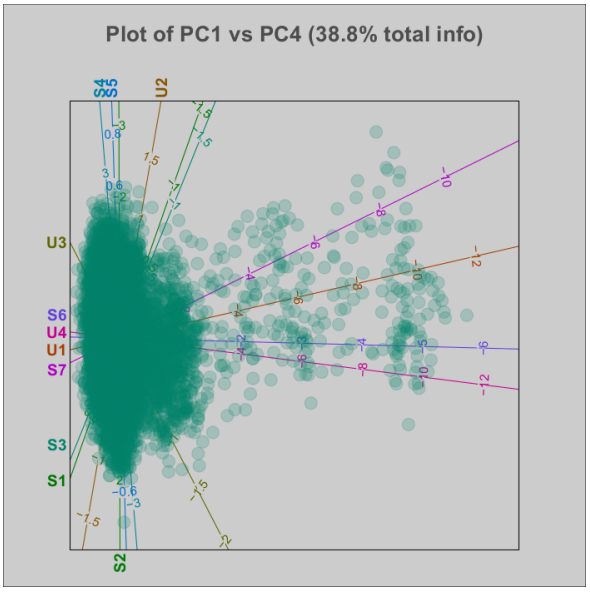
	PC1	PC2	PC3	PC4
U1	-0.54	-0.07	0.02	-0.05
U2	0.12	0.11	-0.63	0.28
U3	-0.19	-0.50	-0.05	0.15
S1	-0.13	0.43	-0.08	-0.15
S2	0.00	-0.15	-0.38	-0.57
S3	-0.15	0.56	0.17	-0.15
S4	-0.14	0.07	0.10	0.70
S5	-0.01	0.25	-0.60	0.16
S6	-0.27	0.34	0.12	0.00
S7	-0.48	-0.15	-0.10	-0.10
U4	-0.54	-0.05	-0.16	0.03



(a) PC1 vs PC2

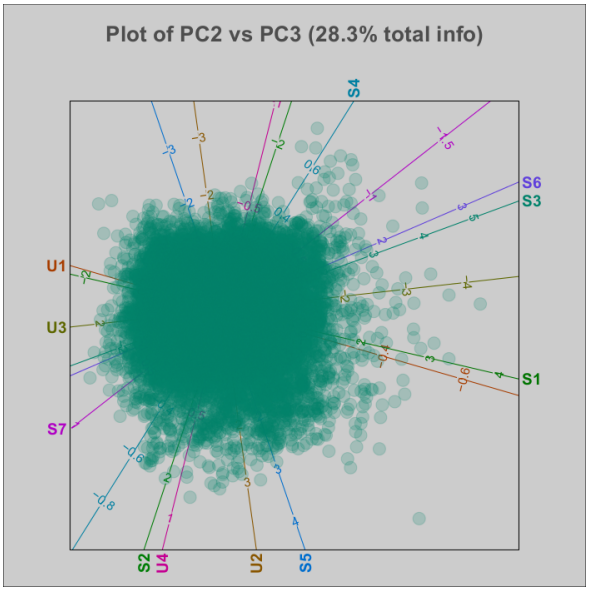


(b) PC1 vs PC3

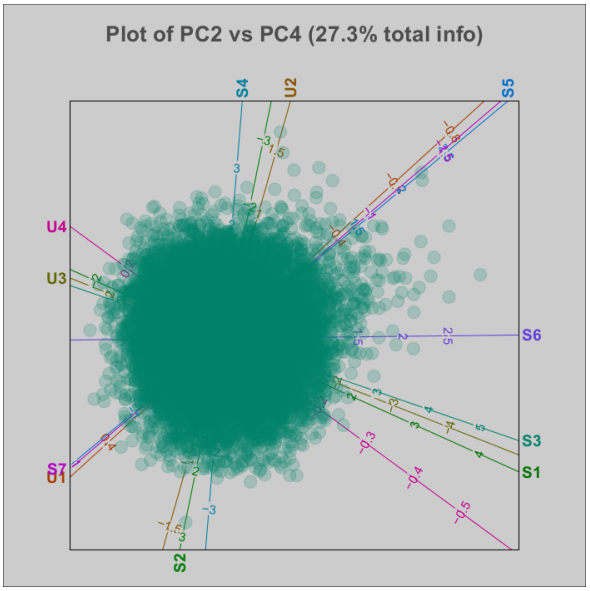


(c) PC1 vs PC4

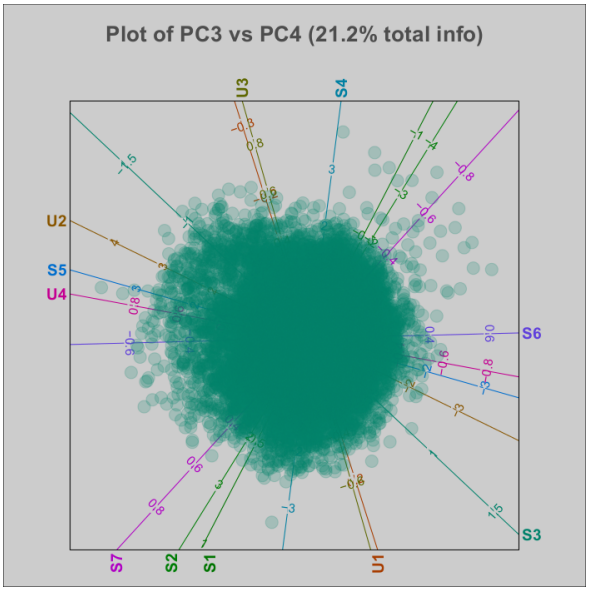
Figure 3.16: PCA plots of different principal component combinations for the reference set



(d) PC2 vs PC3

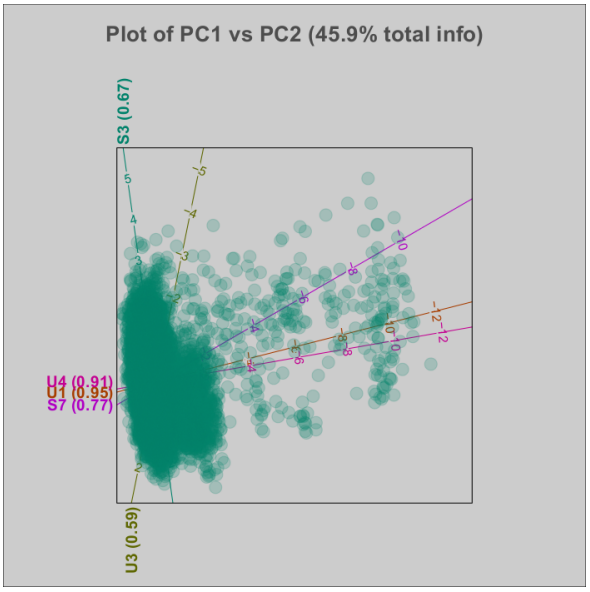


(e) PC2 vs PC4

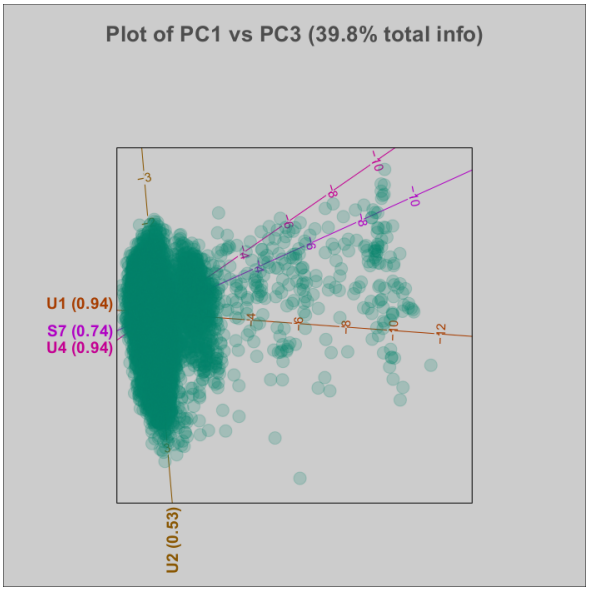


(f) PC3 vs PC4

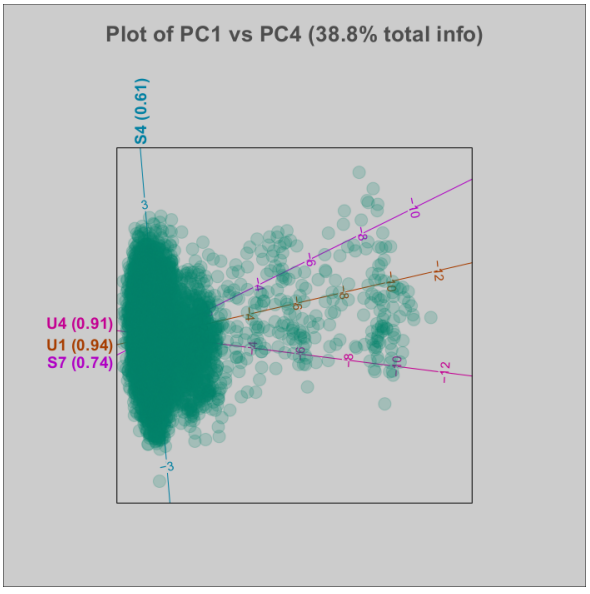
Figure 3.16: PCA plots of different principal component combinations for the reference set continued



(a) PC1 vs PC2

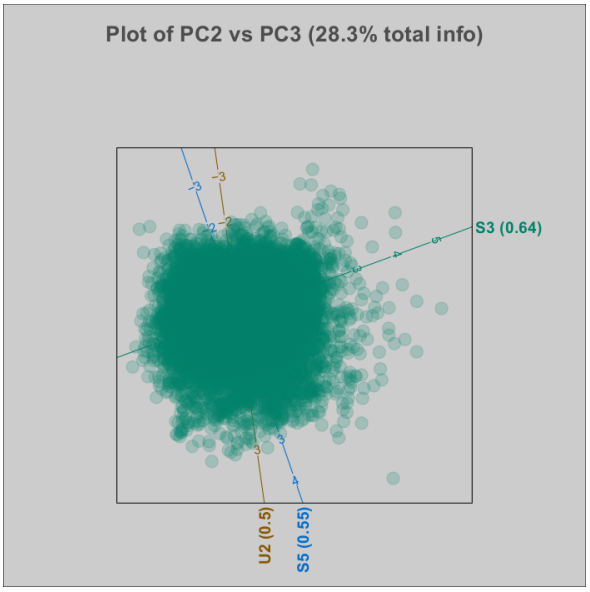


(b) PC1 vs PC3

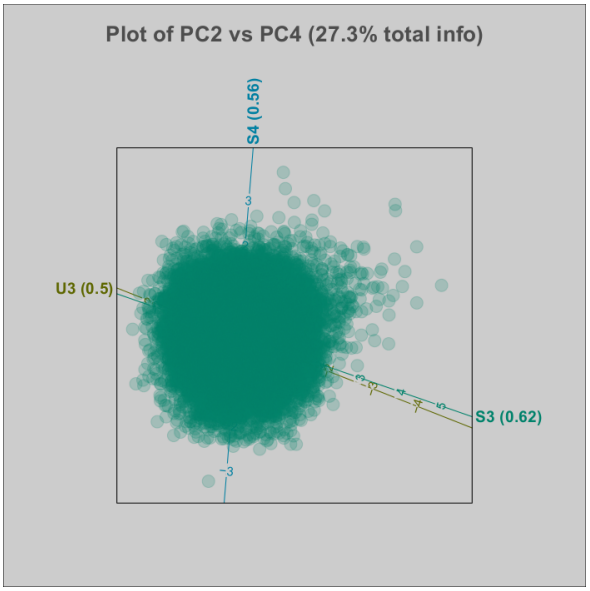


(c) PC1 vs PC4

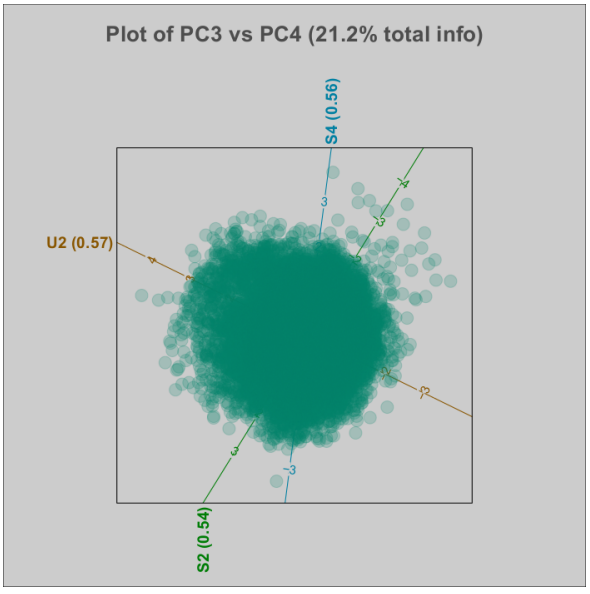
Figure 3.17: PCA plots of different principal component combinations with axes chosen according to the axis predictivity criterion



(d) PC2 vs PC3

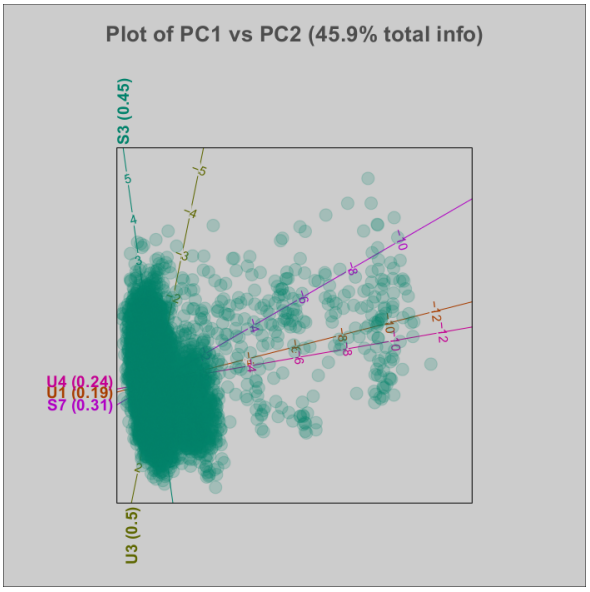


(e) PC2 vs PC4

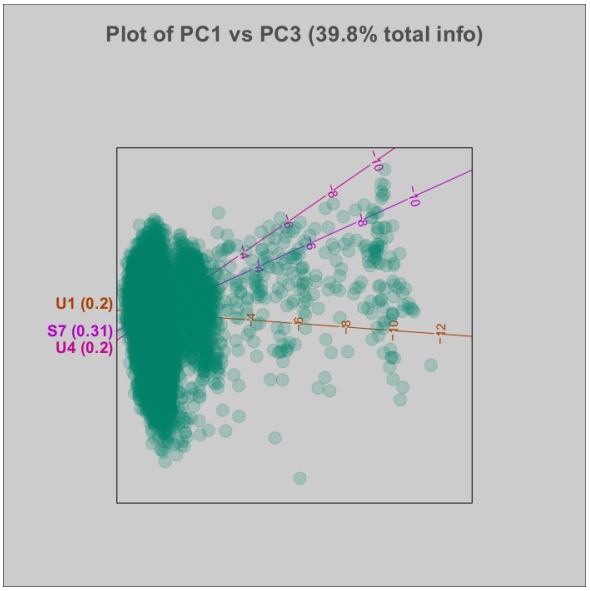


(f) PC3 vs PC4

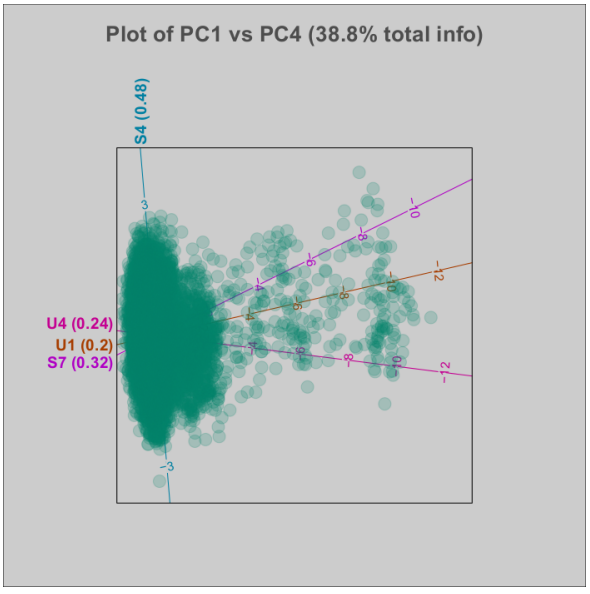
Figure 3.17: PCA plots of different principal component combinations with axes chosen according to the axis predictivity criterion continued



(a) PC1 vs PC2

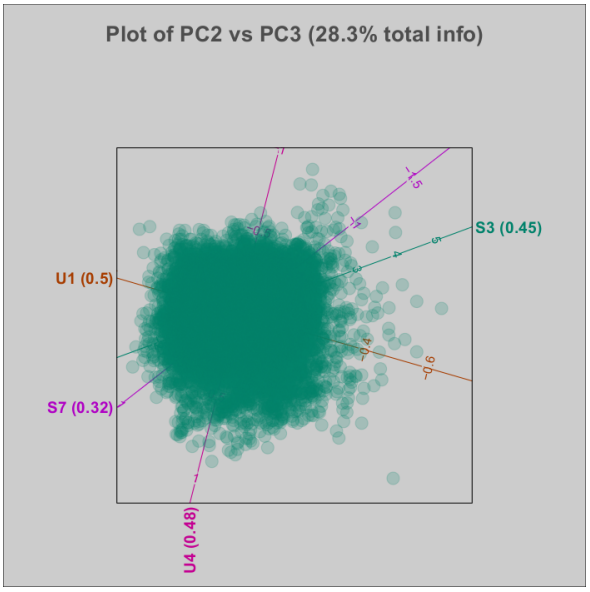


(b) PC1 vs PC3

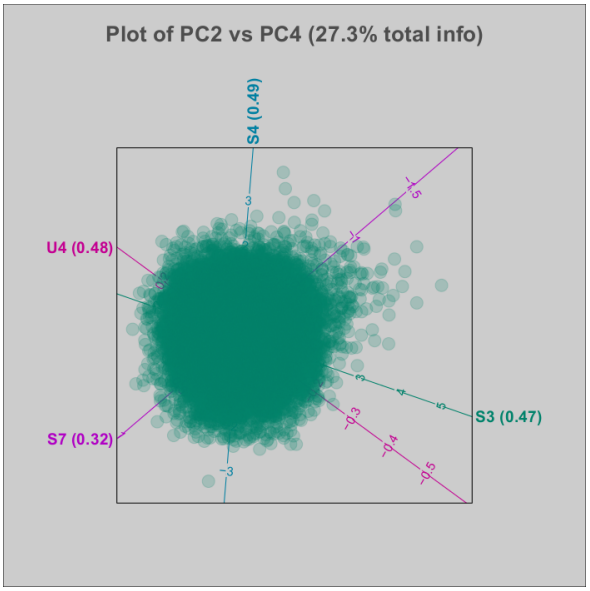


(c) PC1 vs PC4

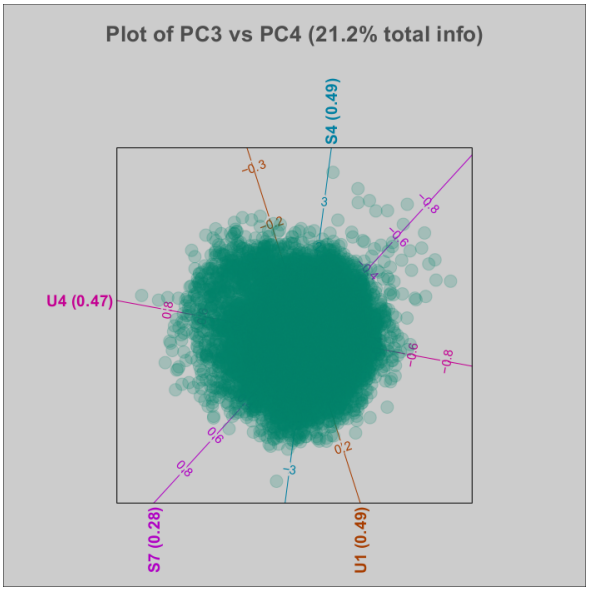
Figure 3.18: PCA plots of different principal component combinations with axes chosen according to the MSPE criterion



(d) PC2 vs PC3



(e) PC2 vs PC4



(f) PC3 vs PC4

Figure 3.18: PCA plots of different principal component combinations with axes chosen according to the MSPE criterion continued

3.3.4.3 Monitoring Biplots

To demonstrate the implementation of monitoring biplots an example of a four hour period captured in Table 3.11 will be discussed. The row numbers indicate the sequence i.e., row one occurred first and row 16 occurred last. Due to confidentiality constraints the data set was centered and scaled using the column means and column standard deviations of the reference set discussed in Section 3.2.4.1. Note that the mean of the reference set will therefore be 0 for all the columns. It is therefore easy to compare the values in Table 3.11 to the reference set as any value above 0 is higher than the average value for the reference set, and any value below 0 is lower than the average value for the reference set. Additionally, the standard deviation of the reference set is scaled to 1, which can also be used to give context to the values in Table 3.11.

Figure 3.19a to Figure 3.19f depicts the data in Table 3.11 projected on the monitoring biplots for the different combinations of PC dimensions discussed in Section 3.3.4.2. The axes included for each dimension have an axis predictivity greater than 0.35 as indicated by the gray cells in Table 3.10.

From Figure 3.19a it can be concluded that points 10 to 12 are outside of the concentration ellipse. Points 10 to 12 project high on variable U3, and low on variables S1 and S3. Considering Table 3.11 it is clear that variable U3 is higher for these points than for the remaining points. Variable S3 is also lower for these points than for the remaining points, and Variable S1 is also in general low for these points. As these points are almost exclusively separated on the PC2 dimension it is expected that they will be separated on all combinations including PC2. This can be confirmed by inspecting Figures 3.19d and 3.19e.

Considering Figure 3.19b it seems that the points moved form 1, 2 and 3 which were outside the concentration ellipse to a grouping of points 4 to 9, which up to the remaining points, were on target for these variables. This separation almost exclusively takes place on the PC3 dimension. Inspecting Table 3.11 it is noted that all these points (1-9) are high for variable U2 (and identical). For variable S5 these points are somewhat higher in general. However, these two variables individually can not explain the separation between points 1-3 and 4-9. Therefore, the remaining plots that contain PC3 are inspected. Figure 3.19d indicates that the points are high on variable U3 and low on variables S3 and S1. Although this can again be confirmed from Table 3.11 these values still do not clearly differentiate between points 1 to 3 and 4 to 9. Figure 3.19f accentuates the separation between these two groups of points. Specifically, points 1-3 project higher on variable S2 and lower on variable S4. This observation can be confirmed from Table 3.11. Points 1-3 are higher for variable S2 than the remaining points, and specifically points 2 and 3 are lower for variable S4 than the remaining points. These results highlight the importance of selecting the appropriate dimensions for out of control detection. The traditional score or biplot of the first two dimensions will not

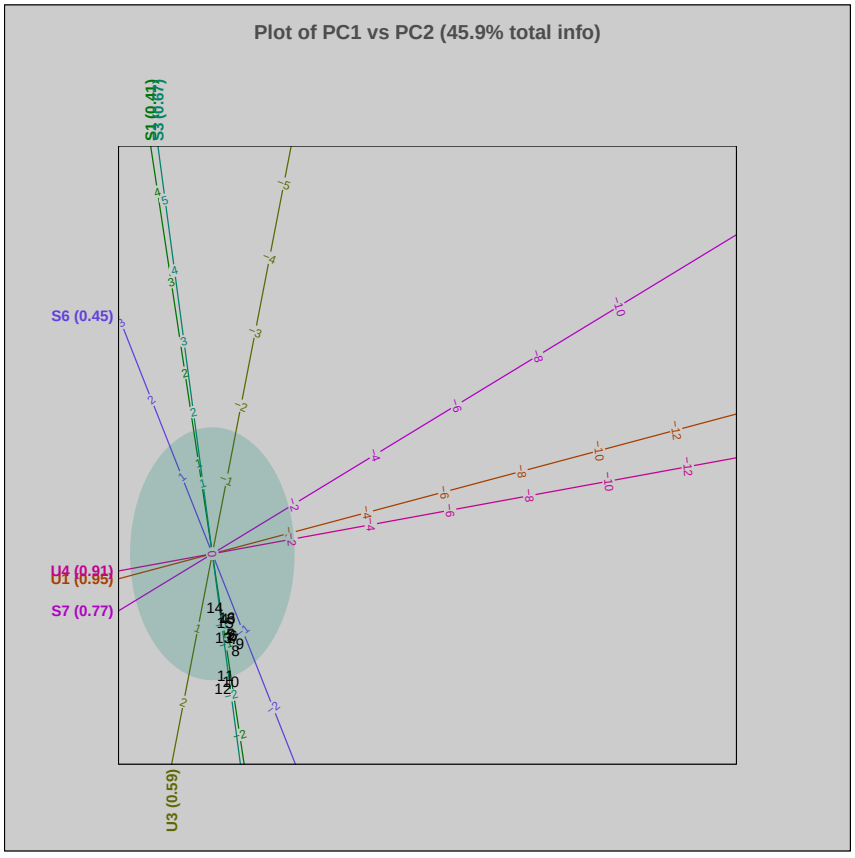
suffice to highlight all the relevant deviations.

Table 3.10: Axis predictivity values with values above 0.35 highlighted

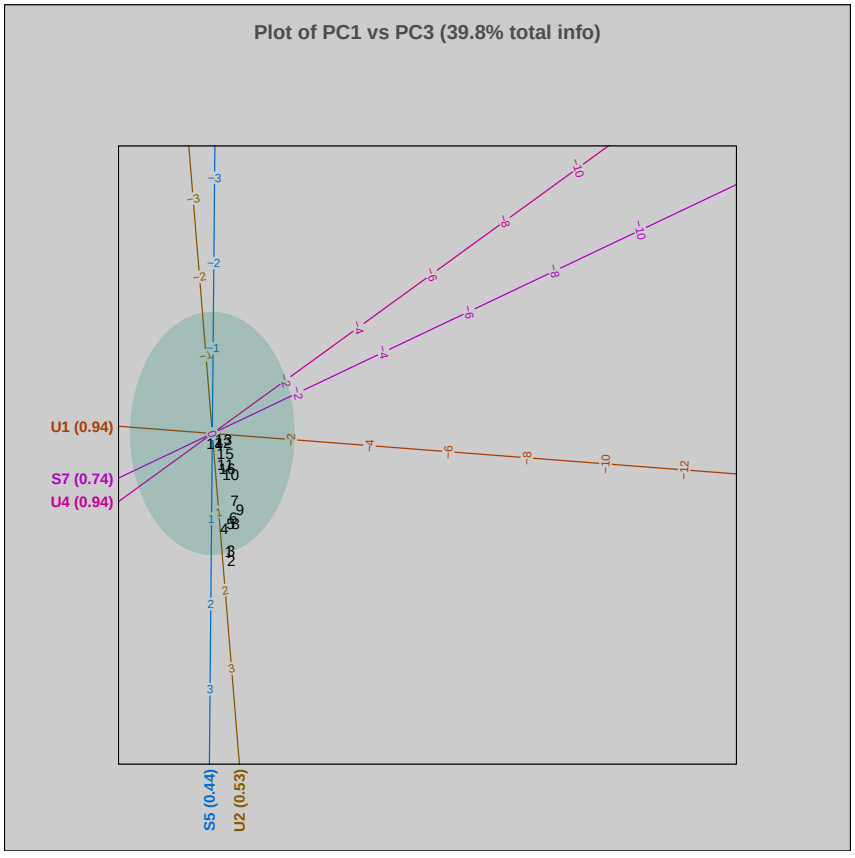
	PC1	PC2	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1	1	2	0.95	0.07	0.59	0.41	0.04	0.67	0.08	0.11	0.45	0.77	0.91
2	1	3	0.94	0.53	0.12	0.06	0.18	0.11	0.08	0.44	0.25	0.74	0.94
3	1	4	0.94	0.14	0.14	0.08	0.36	0.09	0.61	0.03	0.23	0.74	0.91
4	2	3	0.01	0.50	0.48	0.36	0.22	0.64	0.02	0.55	0.24	0.05	0.04
5	2	4	0.01	0.11	0.50	0.38	0.40	0.62	0.56	0.14	0.22	0.05	0.01
6	3	4	0.00	0.57	0.03	0.03	0.54	0.06	0.56	0.47	0.02	0.02	0.03

Table 3.11: Table with new data

	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1	-0.30	1.71	0.64	-0.61	1.08	-2.08	-0.45	0.48	-0.88	0.10	0.21
2	-0.40	1.71	0.89	0.18	1.46	-2.06	-1.54	0.27	-0.83	0.08	0.11
3	-0.34	1.71	0.79	0.64	0.77	-2.01	-1.81	0.29	-0.88	-0.02	0.18
4	-0.22	1.71	0.71	-0.09	-0.04	-1.98	0.13	0.46	-0.95	0.03	0.29
5	-0.31	1.71	0.62	-0.97	-0.27	-1.89	-0.62	0.47	-1.04	0.13	0.20
6	-0.46	1.71	0.54	-1.11	-0.23	-1.95	-0.31	0.32	-1.03	0.22	0.03
7	-0.47	1.71	0.68	-1.07	-0.91	-2.10	-0.15	0.18	-1.12	0.07	0.03
8	-0.51	1.71	0.72	-1.25	-0.04	-2.12	-0.64	0.29	-1.11	0.26	-0.01
9	-0.58	1.71	0.46	-1.22	-0.62	-2.26	-0.66	0.21	-1.10	0.15	-0.08
10	-0.20	0.86	1.27	-1.21	-1.10	-2.91	-0.76	-0.03	-1.13	0.13	0.09
11	-0.09	0.70	1.23	-0.83	-1.01	-2.88	-0.31	-0.23	-1.12	0.19	0.14
12	-0.07	0.70	1.78	-1.48	-1.66	-2.62	-0.19	-0.38	-1.05	0.19	0.15
13	-0.12	0.70	0.74	-1.01	-1.57	-2.05	0.42	-0.19	-0.90	0.07	0.11
14	0.04	0.70	0.55	-0.36	-1.81	-1.81	0.97	0.15	-0.73	0.12	0.26
15	-0.28	0.70	0.32	-1.43	-1.35	-1.62	0.87	0.35	-1.03	0.13	0.14
16	-0.24	0.70	0.17	-0.94	-0.34	-1.35	-0.16	0.03	-1.09	0.12	0.13

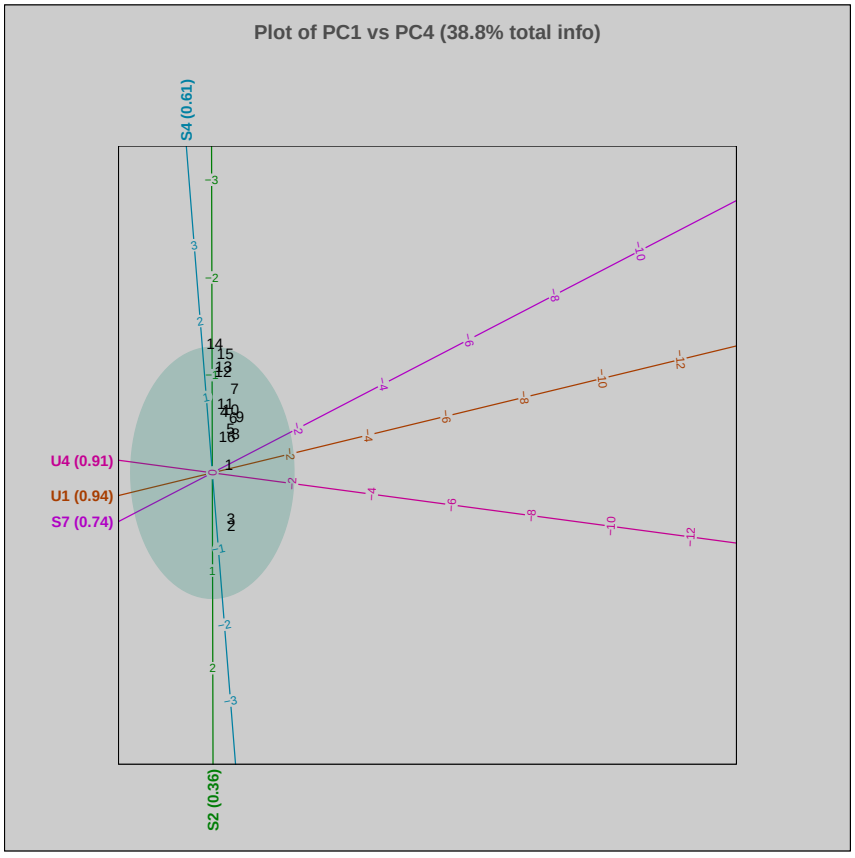


(a) PC1 vs PC2

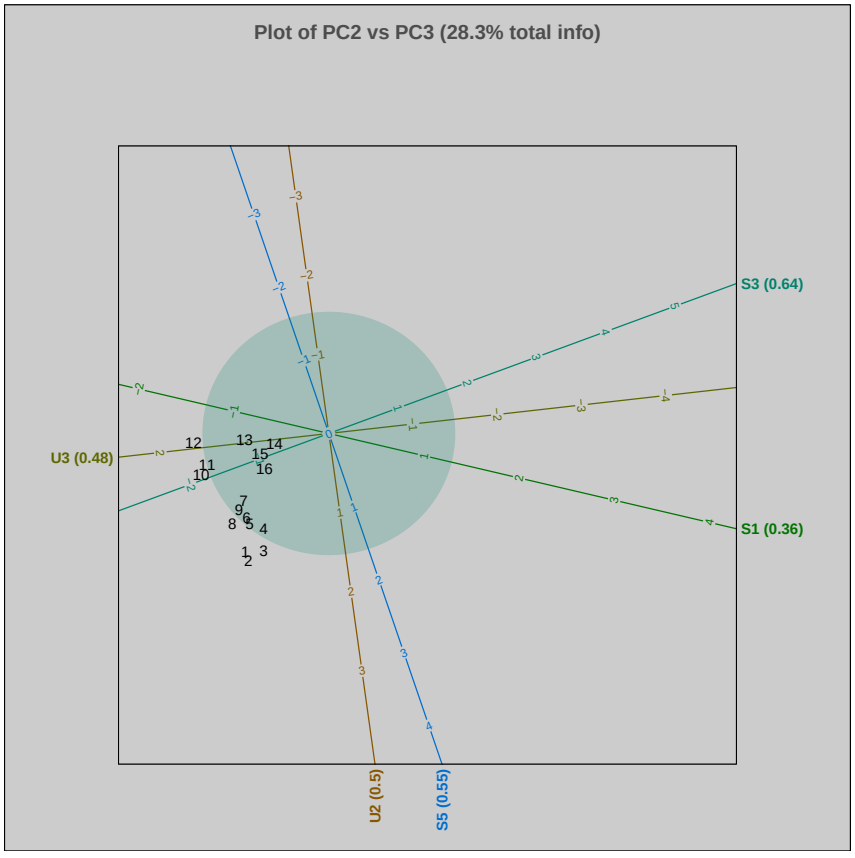


(b) PC1 vs PC3

Figure 3.19: PCA plots of different principal component combinations and monitoring ellipses

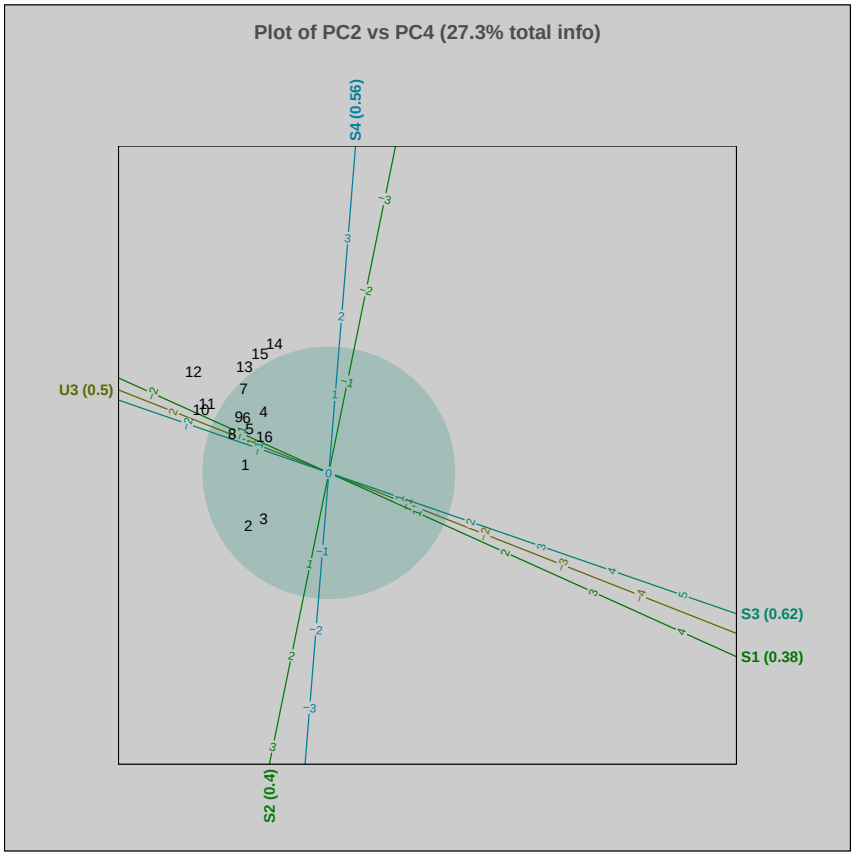


(c) PC1 vs PC4

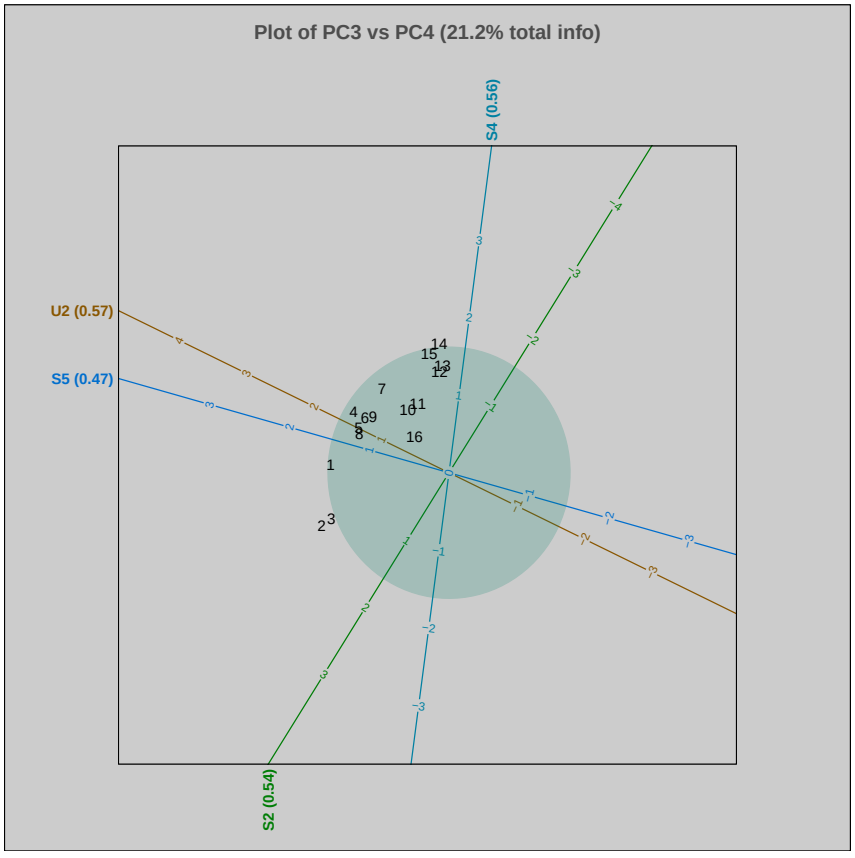


(d) PC2 vs PC3

Figure 3.19: PCA plots of different principal component combinations and monitoring ellipses continued



(e) PC2 vs PC4



(f) PC3 vs PC4

Figure 3.19: PCA plots of different principal component combinations and monitoring ellipses continued

3.3.5 Conclusions for processing monitoring biplots

In this section the PCA biplot and various measures of predictive power of the PCA biplot were investigated, and applied to an industrial case study. Specifically the reference set obtained from the GOPA analysis for the Eastern factory was used (Section 3.2.4.1). The first question that was addressed was the number of eigenvalues (and therefore principal components) to include in the evaluations. Three different measures were compared in the case study:

- The cumulative % variance explained above 80%.
- The eigenvalues of the correlation matrix above one (scree plot).
- The permutation test proposed by Greenacre and Primicerio (2014).

The scree plot indicated that four principal components should be retained. The cumulative % variance explained indicated that five principal components should be retained. The permutation test was performed on both the cumulative % variance explained and the eigenvalues of the correlation matrix. The results clearly indicated that four principal components should be retained. The permutation test is an attractive option as it formalizes the process of principal component selection. It was therefore decided to retain four principal components.

Two measures of predictive power of the biplot axes were applied to the case study. Specifically the mean standard predictive error (mspe) for PCA biplots from Alves (2012) and the PCA axis predictivities from Gardner-Lubbe *et al.* (2008) were implemented and applied. In this case study, similar results were obtained using these two measures. However, the results from the PCA axis predictivities could be explained from a process perspective, and was investigated further.

From the axis predictivity results it was concluded that in addition to the intended axes allocation on the different biplots, the predictivity values on their own contains valuable information. Careful analysis of the axis predictivity highlights the variables captured by the different principal components. It could be illuminating to investigate the different groupings on the principal components for hidden interactions. The groupings contained in the case study were validated against the underlying process, and found to be a true reflection of the underlying process dynamics.

Concentration ellipses were added to the selected principal component and axes combinations, and a new data set containing a four hour period of fifteen minute average data was projected onto the biplots. From this application it was concluded that more information is obtained by only including the axes with high predictive power, and by not only investigating the traditional PC1 $\times$ PC2 combination. It was also clear that a number of the PC combinations can be useful to extract the maximum information from the underlying structure of the data.

Therefore it is suggested that in any PCA biplot analysis the proposed methodology should be employed. This is especially important for the real time on-line application of this case study. Information on the PCA biplot quality and the axis predictivities should be supplied on the plots to indicate to the user the confidence in the results. It is also prudent to compare the results obtained to the actual data to confirm the results. It should however be noted that the actual process consists of 84 gasifiers, and it would be an overwhelming number of biplots to analyze if the user is not guided to the important gasifiers in a constructive manner. A solution to this problem will be proposed in Chapter 4. No part of the biplot analysis stands alone, and therefore the steps in Figure 3.20 were suggested for the implementation of a PCA biplot monitoring methodology for multiple identical production processes.

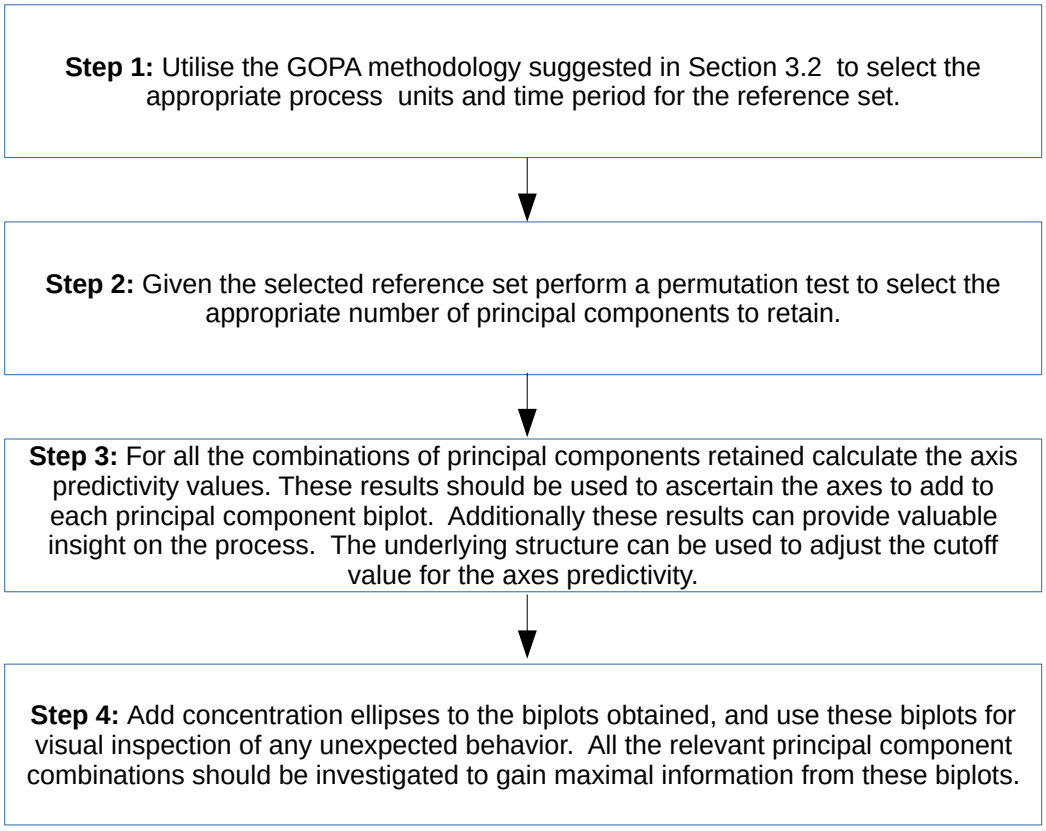


Figure 3.20: Steps for the implementation of a PCA biplot monitoring methodology

In conclusion, PCA biplots are powerful visual aids in multivariate analysis. A significant amount of information can be obtained in one glance that would be virtually impossible to obtain from tables of data. It is however necessary to exclude axes with low axis predictivity values as to not misguide the user, as the casual user will assign equal importance to all the axes as a direct consequence

of the comparison of scatter plots to biplots often made in the literature. In addition, more eigenvalues than the first two should be investigated as there is possibly additional separation taking place on the remaining planes orthogonal to the first two principal components. The code used to create the graphs in this study will be discussed in detail in Section 5.6.1. The PCA biplots discussed in this section are appropriate for the real-time monitoring of unexpected behavior over short time periods. In the next section CVA biplots will be discussed for longer term (but still real-time) monitoring of the differences between the different process units on a specific train.

3.4 CVA Biplots for Monitoring

3.4.1 Introduction

In this study the monitoring of processes involving several externally defined groups such as different plants, or reactors is considered. Specifically, canonical variate analysis (CVA) biplot methodology is applied for monitoring these types of processes. The use of the CVA biplot as a monitoring graph has been discussed previously by Aldrich *et al.* (2004). In this study the predictive power of the biplot axes is investigated in different dimensions by extending the ideas introduced by Alves (2012) for principal component analysis (PCA) biplots to CVA biplots. Various types of axis predictivities have been defined previously by Gardner-Lubbe *et al.* (2008) and Gower *et al.* (2011) for CVA biplots. The usage and the relative merits of these CVA axis predictivities, and an extended measure for monitoring multivariate processes with external group structure, are investigated. An R (R Core Team, 2015) function is presented for automating the monitoring methodology. The methodology will be presented using the same coal gasification case study. The case study is discussed in Section 3.1.

3.4.2 Problem setting

The current facility under investigation consists of 10 production processes (reactors) grouped into one production train. In this study data from one of the production trains will be used. Although this is only one section of the full production process, it contains all the complexities of the overall process. In the current study, time weighted hourly average data for twelve process variables were captured over a consecutive one week period for all 10 processes. The process variables include measurements on utilization such as oxygen and steam consumption and other stability measurements on the reactors. The variables are identical to the variables discussed in the previous section, listed in Table 3.1. Note that the variables are described by neutral labels due to confidentiality restrictions imposed by the company under consideration.

During the day to day monitoring of the facility, CVA biplots are used to detect any deviation of the average performance of a reactor from the average performance of all the reactors on the train. Each reactor on a train receives the same feedstock, and is operated by the same operator. Therefore, any performance deviation of a reactor from the other reactors on a train needs to be investigated as it could indicate imminent mechanical breakdown and consequently loss of production.

An example of a typical CVA biplot used for monitoring the processes is provided in Figure 3.21. The plot was created using a custom biplot package developed as part of this study. The biplot package will be discussed in detail in Chapter 5. In addition to the reactor means the 90% alpha bags are dis-

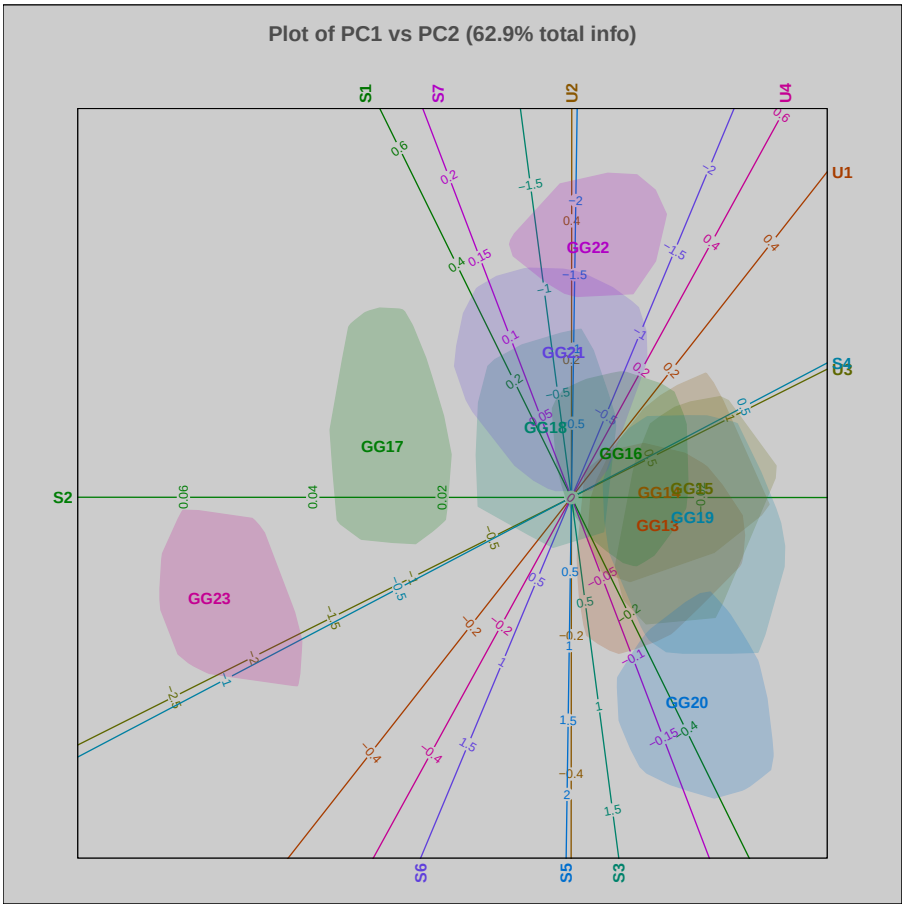


Figure 3.21: CVA Biplot of a production train

played on the plot for the reactors. As described by Gower *et al.* (2011), these bags enclose the inner most 90% of the data points for each of the reactors. The bagplot is a bivariate generalisation of the boxplot (Aldrich *et al.*, 2004). The CVA biplots contain a rich set of information. For example, from the plot it is observed that the performance of gasifiers GG17 and GG23 differ significantly from the remaining reactors on the train. In addition to which reactors differ from the rest, the variables contributing to the differences are also suggested from the plots by projecting the means onto each axis. Furthermore, variation in the performance of all reactors on any of the 11 variables is visually portrayed in the CVA biplot.

In the current monitoring work flow the data are gathered in real time, and the biplots are generated automatically and displayed on a website. The plots are part of a larger web based monitoring package which was developed in-house. Even though the real-time aspect of the plots is very beneficial it also adds complexity to the problem. In a normal offline statistical analysis using biplots the analyst can scrutinize the plots, and ensure that the results are statistically sound. In the real-time or on-line scenario this is not possible,

and this can lead to misinterpretation of the plots leading to invalid conclusions if the necessary care is not taken.

Two specific issues will be addressed in this study. The first issue is the overall quality of the CVA biplot, and the closely related combination of principal axes to use as scaffolding for constructing the plots. In the web based interface, two-dimensional biplots are preferred and three-dimensional plots will therefore not be discussed in this study. In addition to the combination of the principal axes used as scaffolding for the biplot, the predictive power of the calibrated axes (i.e. the representation of the variables) in each plot will be addressed. The predictive power of the axis is very important in that firstly a large number of axes on the plot makes it very difficult to read, and additionally having an axis with very low predictive power present on a plot can lead to wrong conclusions being made from the variable contributions.

As an example, from Figure 3.21 it can be concluded that gasifier GG23 has relatively high values of variables S2 and S6 and low values for S3, S4, U1 and U4. It is however possible that some of the respective biplot axes have very low predictive power and can be virtually orthogonal to the current plane. The axis direction could therefore be almost arbitrary, rendering them almost useless for valid conclusions. Additionally it is possible that although the reactors in the center are performing similar in this plane, in a different dimension they could be separated. The CVA biplot code demonstrated in this section allows the analyst to investigate these questions, and choose the axes to display as well as the principal axes to use for the scaffolding on which to construct the whole biplot. Moreover, in this application the automatic real-time aspect also needs to be addressed. In the next section the underlying mathematics of the CVA biplot as well as some measures of the predictive power of the plots will be discussed.

3.4.3 CVA Biplot

The two-stage approach to the CVA biplot as discussed in Gower *et al.* (2011) is convenient for developing our monitoring procedure. Therefore, in this section a brief description of essential ideas underlying this two-stage approach is presented. Consider an $n \times p$ data matrix $\boldsymbol{X}$ consisting of J groups. The group sizes are $n_1, \cdots, n_J$, such that $\sum_{i=1}^{J} n_i = n$. Also, p variables are measured for the n_i samples in each of the J groups. Furthermore let $\boldsymbol{G}$ be an $n \times J$ indicator matrix with unity in the j_ith entry of $\boldsymbol{G}$ for all n_i in group J_i and 0 otherwise. Furthermore, let $\boldsymbol{N}$ be a $J \times J$ diagonal matrix with $n_1, \cdots, n_J$ in the diagonal positions. It is assumed that $\boldsymbol{X}$ is mean centered, i.e the mean of each column is 0.

The $J \times p$ matrix of group means can therefore be calculated as

$$\bar{\boldsymbol{X}} = \boldsymbol{N}^{-1}\boldsymbol{G}^T\boldsymbol{X} \tag{3.4.1}$$

Let the total ($\boldsymbol{T}$), within ($\boldsymbol{W}$) and between ($\boldsymbol{B}$) sums of squares and products (SSP) matrices be denoted by

$$\boldsymbol{T} = \boldsymbol{X}^T\boldsymbol{X} \tag{3.4.2}$$

$$\boldsymbol{W} = \boldsymbol{X}^T\boldsymbol{X} - \bar{\boldsymbol{X}}^T\boldsymbol{N}\bar{\boldsymbol{X}} \tag{3.4.3}$$

$$\boldsymbol{B} = \bar{\boldsymbol{X}}^T\boldsymbol{N}\bar{\boldsymbol{X}} \tag{3.4.4}$$

i.e. $\boldsymbol{T} = \boldsymbol{B} + \boldsymbol{W}$.

The first step in the two-stage process is to find a transformation of the variables such that the Pythagorean distances between the group means of the transformed variables are equal to the Mahalanobis distances between the untransformed group means of the variables. Therefore a $p \times p$ matrix $\boldsymbol{L}$ is required where $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{W}^{-1}\boldsymbol{x}$. This implies that $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{W}^{-1}$, and therefore $\boldsymbol{L}^T\boldsymbol{W}\boldsymbol{L} = \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix.

Solving the eigenvector equation

$$\boldsymbol{W}\boldsymbol{L} = \boldsymbol{L}\boldsymbol{\Lambda} \tag{3.4.5}$$

where the eigenvectors are scaled to give $\boldsymbol{L}^T\boldsymbol{W}\boldsymbol{L} = \boldsymbol{I}$ provides the transformation matrix $\boldsymbol{L}$.

The transformed variables $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{L}$ are called the canonical variables. The transformed group means $\bar{\boldsymbol{X}}\boldsymbol{L}$ are the means of the canonical variables and are called the canonical means. Furthermore,

$$\bar{\boldsymbol{X}}\boldsymbol{L}\boldsymbol{L}^T\bar{\boldsymbol{X}}^T = \bar{\boldsymbol{X}}\boldsymbol{W}^{-1}\bar{\boldsymbol{X}}^T = \bar{\boldsymbol{X}}(\boldsymbol{X}^T\boldsymbol{X} - \bar{\boldsymbol{X}}^T\boldsymbol{N}\bar{\boldsymbol{X}})^{-1}\bar{\boldsymbol{X}}^T \tag{3.4.6}$$

implying that in the canonical space the Pythagorean distance between the canonical means ($\bar{\boldsymbol{X}}\boldsymbol{L}\boldsymbol{L}^T\bar{\boldsymbol{X}}^T$) are equal to the Mahalanobis distance between

the group means in the original observation space ($\bar{\boldsymbol{X}}\boldsymbol{W}^{-1}\bar{\boldsymbol{X}}^T$). This concludes the first step of the two-staged approach. An important consequence of the transformation is that the weighted sum of the rows of $\boldsymbol{X}$ vanishes, therefore the points given by the J rows of $\bar{\boldsymbol{X}}\boldsymbol{L}$ will occupy at most $\min(J-1, p)$ dimensions of the canonical space.

In the second step Gower *et al.* (2011) introduced a PCA to be performed on the canonical means. In CVA the deviations from the group means can be weighted by the group sizes. The unweighted Pythagorean distances between the canonical means on the left hand side of (3.4.6) can be extended to provide for weighted by group size distances by replacing the implicit identity matrix with a $J \times J$ matrix $\boldsymbol{C}$, where for the weighted case $\boldsymbol{C} = \boldsymbol{N}$, and for the unweighted case $\boldsymbol{C} = \boldsymbol{I} - J^{-1}\mathbf{1}\mathbf{1}^T$.

The eigenvalue decomposition can then be written as

$$(\boldsymbol{L}^T\bar{\boldsymbol{X}}^T\boldsymbol{C}\bar{\boldsymbol{X}}\boldsymbol{L})\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda} \tag{3.4.7}$$

The matrix $\boldsymbol{V}$ is the usual PCA orthogonal transformation matrix from the canonical variables. It is however convenient to apply the transformation to the canonical variables and the PCA transformation directly to the original variables. This can be achieved by defining the transformation matrix $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{V}$. If the $p \times p$ matrix $\boldsymbol{K}$ is defined with all 0's except on the diagonal which consists of 1's if a dimension is included and 0's otherwise, the approximation in lower dimensions can be defined as

$$\bar{\boldsymbol{X}}\boldsymbol{L}\boldsymbol{V}\boldsymbol{K} = \bar{\boldsymbol{X}}\boldsymbol{M}\boldsymbol{K} \tag{3.4.8}$$

The PCA prediction of the original group means can be found by

$$\hat{\bar{\boldsymbol{X}}} = \bar{\boldsymbol{X}}(\boldsymbol{M}\boldsymbol{K}\boldsymbol{M}^{-1}). \tag{3.4.9}$$

3.4.3.1 Predictive Measures for CVA Biplots

Several measures for the predictive power of CVA biplots have been proposed in literature (Gardner-Lubbe *et al.*, 2008; Gower *et al.*, 2011; Gower and Hand, 1996). Additionally Alves (2012) proposed an alternative measure for PCA biplots. In this section the measures for predictive power of CVA biplots axes will be listed, and the measure proposed by Alves (2012) will be extended to CVA biplots.

Due to the second step of the two-stage process being a normal PCA transformation the quality of the biplot in the canonical variables could be obtained as for a PCA analysis, but generally the quality in the original variables is of interest, and not so much the quality in the canonical variables. Therefore, the focus will be on the original variables in this study. Overall quality in the original variables, axis adequacy and axis predictivity are discussed extensively

in Gower *et al.* (2011) and Gardner-Lubbe *et al.* (2008) and are summarized below.

$$\text{Overall Quality} = \frac{\text{tr}(\hat{\bar{\boldsymbol{X}}}^T \boldsymbol{C} \hat{\bar{\boldsymbol{X}}})}{\text{tr}(\bar{\boldsymbol{X}}^T \boldsymbol{C} \bar{\boldsymbol{X}})}. \tag{3.4.10}$$

Axis adequacy is defined as the diagonal elements of the $p \times p$ matrix

$$\text{diag}(\boldsymbol{M}\boldsymbol{K}\boldsymbol{M}^T)[\text{diag}(\boldsymbol{M}\boldsymbol{M}^T)]^{-1}. \tag{3.4.11}$$

Axis predictivity is defined as the diagonal elements of the $p \times p$ matrix $\boldsymbol{\Pi}$ where

$$\boldsymbol{\Pi} = \text{diag}(\hat{\bar{\boldsymbol{X}}}^T \boldsymbol{C} \hat{\bar{\boldsymbol{X}}})[\text{diag}(\bar{\boldsymbol{X}}^T \boldsymbol{C} \bar{\boldsymbol{X}})]^{-1}. \tag{3.4.12}$$

The mean standard prediction error (mspe) discussed in Alves (2012) is specific for PCA biplots, but can be adapted for CVA biplots as

$$\text{mspe} = \frac{\frac{1}{J}\mathbf{1}^T|\bar{\boldsymbol{X}} - \bar{\boldsymbol{X}}(\boldsymbol{M}\boldsymbol{K}\boldsymbol{M}^{-1})|}{\boldsymbol{\sigma}^T} = \frac{\frac{1}{J}\mathbf{1}^T|\bar{\boldsymbol{X}} - \hat{\bar{\boldsymbol{X}}}|}{\boldsymbol{\sigma}^T} \tag{3.4.13}$$

where $\mathbf{1}$ is a $1 \times J$ vector of 1's and $\boldsymbol{\sigma}$ is the vector of column standard deviations of $\bar{\boldsymbol{X}}$

3.4.4 Results

First the number of eigenvalues that needs to be included in the biplot analysis is determined. As discussed in section 3.4.3 the canonical variables will occupy at most $\min(J-1, p)$ dimensions of the canonical space. In this study there are 10 groups, and 11 variables, therefore the canonical variables can be fully represented in 9 dimensions. Alves (2012) discussed various methodologies to determine the number of eigenvalues to include in the study. It can be observed from the scree plot in Figure 3.22 that the first three eigenvalues are larger than one. In Figure 3.23 the cumulative percentage variance explained is depicted. This is the variance explained in the original variables utilising equation (3.4.10). The first four eigenvalues explain more than 80% of the total variance. It was decided to use four eigenvalues in this study.

CVA biplots were constructed for all $\binom{4}{2} = 6$ combinations of eigenvalues. In Table 3.12 the biplot quality in the original variables is provided. The biplot quality for the third against the fourth eigenvalues is very low, but it could still be instructive to evaluate the information provided.

In Tables 3.13 and 3.14 the axis predictivities from (3.4.12) and the mean standard prediction errors from (3.4.13) are provided. In Table 3.13 the axes with a minimum axis predictivity of 0.4 are highlighted and similarly in Table 3.14 the axes with a maximum mspe value of 0.6 are highlighted. It is informative to note that the results are very similar. There are some differences on the axes included, but these are for axes that are very close to the specified

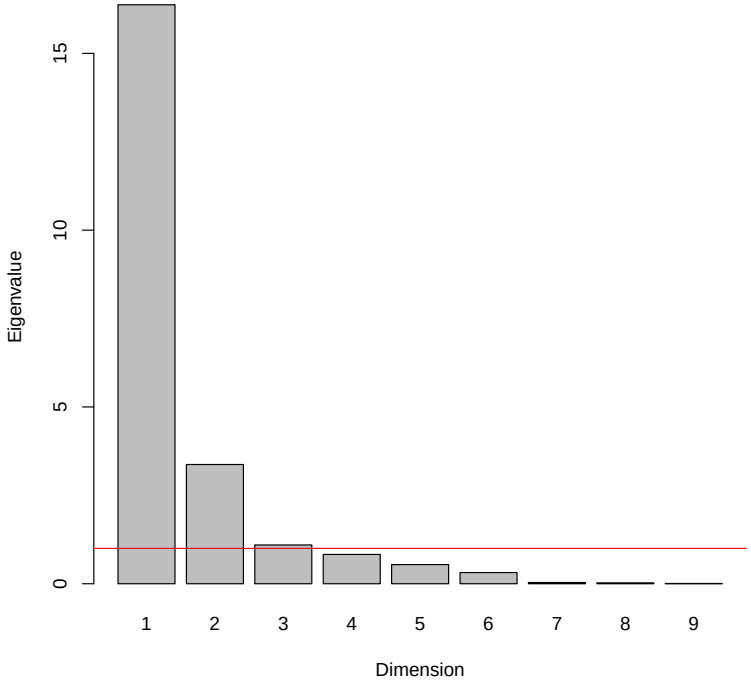


Figure 3.22: Scree plot

Table 3.12: CVA biplot quality in the original variables for Figures 3.24a to 3.24f

	PC (x-axis)	PC (y-axis)	Quality (%)
Figure 3.24a	1	2	62.91
Figure 3.24b	1	3	45.35
Figure 3.24c	1	4	49.55
Figure 3.24d	2	3	33.37
Figure 3.24e	2	4	37.56
Figure 3.24f	3	4	20.01

values i.e., for axis U1 three values are inside the axis predictivity limits, and none for the mspe limit.

As biplot axis predictivity and mean standard prediction error lead to similar results, it was decided to use axis predictivity to generate the plots. In Figures 3.24a to 3.24f the CVA plots are depicted with the different combinations of eigenvalues. In addition to the axes names, the predictivity for each axis is also provided in brackets. The principal component (PC) combinations as well as the total percentage variation explained (in the original variables) are provided in the plot titles. The first principal component in the title is

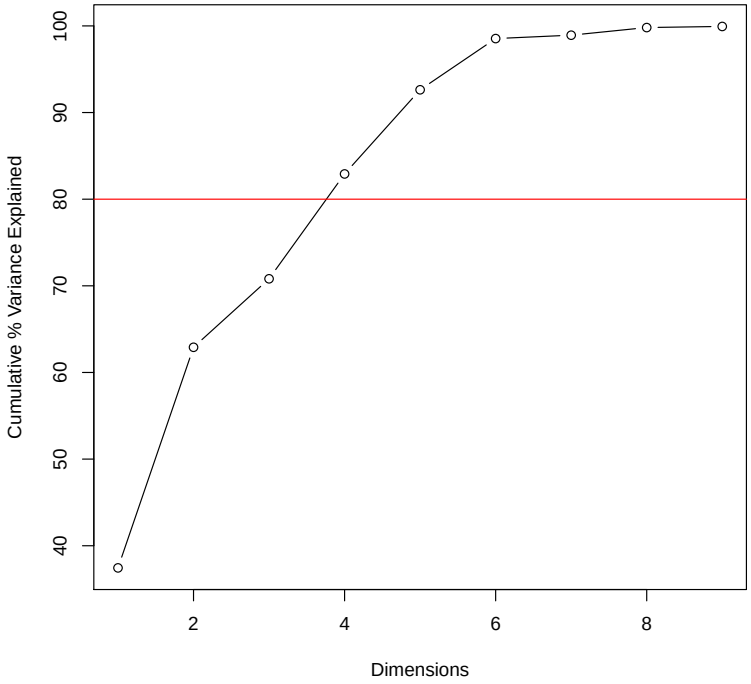


Figure 3.23: Cumulative percentage variance explained

Table 3.13: Axis predictivity values

PC	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1×2	0.26	0.07	0.98	0.55	0.01	0.64	0.33	0.73	0.79	0.12	0.35
1×3	0.47	0.02	0.96	0.53	0.21	0.10	0.40	0.03	0.69	0.14	0.55
1×4	0.44	0.49	0.96	0.39	0.05	0.09	0.81	0.00	0.49	0.30	0.37
2×3	0.29	0.08	0.03	0.31	0.20	0.56	0.09	0.76	0.49	0.12	0.37
2×4	0.26	0.56	0.03	0.17	0.04	0.55	0.50	0.73	0.29	0.27	0.20
3×4	0.47	0.51	0.00	0.14	0.24	0.02	0.57	0.03	0.20	0.30	0.39

Table 3.14: Axes MSPE values

PC	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1×2	0.67	0.68	0.12	0.49	0.73	0.50	0.60	0.32	0.42	0.63	0.64
1×3	0.62	0.70	0.20	0.54	0.68	0.74	0.67	0.58	0.49	0.72	0.58
1×4	0.63	0.61	0.20	0.64	0.81	0.77	0.34	0.60	0.61	0.78	0.63
2×3	0.66	0.73	0.69	0.55	0.68	0.56	0.77	0.29	0.54	0.68	0.59
2×4	0.65	0.58	0.70	0.65	0.80	0.55	0.49	0.31	0.57	0.74	0.69
3×4	0.67	0.64	0.71	0.70	0.75	0.75	0.46	0.58	0.72	0.76	0.67

Table 3.15: Group mean values

	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
GG13	0.16	-0.06	0.60	-0.03	0.06	0.17	-0.32	0.06	-0.08	0.06	0.14
GG14	0.23	-0.18	0.38	-0.12	0.18	0.23	0.20	0.21	-0.91	0.13	0.21
GG15	0.12	0.01	0.45	-0.06	-0.15	0.40	1.74	-0.34	-0.01	0.03	0.13
GG16	0.34	-0.98	0.46	-0.07	-0.04	-0.76	0.51	-0.16	-0.43	0.07	0.17
GG17	0.25	0.17	-1.15	-0.12	0.17	0.10	-0.54	-0.38	-0.15	0.22	0.28
GG18	0.33	-0.23	0.02	0.29	0.34	-0.38	-0.04	-0.46	-0.20	0.23	0.29
GG19	-0.02	0.50	0.72	-0.41	0.05	0.62	-0.20	-0.48	-0.08	-0.05	0.06
GG20	-0.14	-0.13	0.35	-0.33	-0.15	0.94	-0.14	1.66	0.28	-0.34	-0.18
GG21	-0.63	1.25	0.21	0.38	-0.50	-1.20	-0.37	-0.21	-0.67	-0.44	-0.42
GG22	-0.03	-0.34	0.57	0.39	0.08	-0.61	-0.51	-3.29	-0.23	0.15	-0.06
GG23	-0.70	-0.17	-2.11	0.44	-0.03	-0.13	-0.80	0.63	2.21	0.04	-0.72

Table 3.16: Group standard deviation values

	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
GG13	0.72	0.99	0.51	1.00	0.87	0.81	0.67	0.23	0.80	0.73	0.70
GG14	0.77	0.12	0.40	0.85	0.88	0.66	0.72	0.72	0.55	0.86	0.79
GG15	0.51	0.60	0.42	1.00	0.80	0.80	0.47	0.77	0.71	0.47	0.53
GG16	0.44	0.84	0.43	0.87	0.83	0.81	0.64	0.57	0.42	0.06	0.40
GG17	0.46	0.96	0.32	0.83	0.91	0.77	0.63	1.01	0.49	0.27	0.46
GG18	0.40	0.12	0.37	1.03	1.00	0.71	0.71	0.50	0.71	0.14	0.41
GG19	0.54	1.15	0.58	0.94	0.97	0.80	0.90	0.52	0.45	0.52	0.56
GG20	1.37	0.63	0.58	0.85	0.95	0.64	0.89	0.95	0.58	1.65	1.37
GG21	1.89	1.45	0.91	1.33	1.20	1.08	1.02	0.45	0.72	2.25	1.95
GG22	0.62	0.15	0.76	1.06	1.35	0.96	0.53	0.15	0.91	0.17	0.64
GG23	1.22	0.82	0.34	0.88	1.24	0.87	0.66	0.42	0.35	0.74	1.22

Table 3.17: CVA classification accuracy

PC	GG13	GG14	GG15	GG16	GG17	GG18	GG19	GG20	GG21	GG22	GG23
1×2	0.43	0.19	0.48	0.61	0.99	0.65	0.34	0.98	0.64	0.96	0.99
1×3	0.25	0.54	0.60	0.49	0.99	0.63	0.43	0.38	0.30	0.68	0.98
1×4	0.34	0.35	0.72	0.61	0.99	0.75	0.59	0.37	0.38	0.32	0.96
2×3	0.31	0.47	0.59	0.19	0.58	0.31	0.11	0.87	0.79	0.89	0.69
2×4	0.32	0.38	0.47	0.54	0.20	0.43	0.41	0.88	0.68	0.96	0.74
3×4	0.31	0.29	0.63	0.53	0.68	0.31	0.40	0.16	0.06	0.82	0.40

plotted on the x axis, and the second principal component in the title on the y axis. Therefore in Figure 3.24a PC2 will be on the y axis, and in Figure 3.24d PC2 will be on the x axis. Additionally in Tables 3.15 and 3.16 the group means and standard deviations are provided for validation purposes.

In these CVA plots the individual samples are not interpolated on the plots, only the group means are depicted. This decision was made to aid interpretation, as the samples will clutter the display. However, the 90% alpha bags were added to the plots to give some indication of the spread of the samples. It is informative to note that in PCA biplots the samples are used to

create the plot and the group means are interpolated onto the plot, whereas in CVA biplots the group means are used to create the plots, and the individual samples are interpolated onto the plot.

CVA biplots provide for the classification of samples by classifying a sample as belonging to the same group as the closest group mean value using Euclidean distances in the canonical space. All n samples were classified using this methodology, and in Table 3.17 the proportion of samples belonging to a group that was classified as belonging to the group is provided. It can be concluded that reactors with high classification accuracy are performing significantly different from the remaining reactors.

Referring to Table 3.13 some patterns in the data are apparent. The patterns are less clear than in the PCA analysis (Section 3.3.4.2). However, it remains evident that variable U3 has the highest loadings on the first principal component. Variable U2 and S4 have the highest loadings on the fourth principal component. Similarly S3 and S5 are mostly represented by the second principal component.

For the implementation of this work, new CVA biplot code was developed in the `R` software (R Core Team, 2015). The `R` code is provided in Appendix B. Comparing Figure 3.24a to Figure 3.21, it is clear that the plots are identical. Note that the centering and scaling aid in the interpretation of the data since all the variables are represented on the same scale (mean equal to 0 and standard deviation equal to 1). In addition, only five of the axes are present on the plot, and it is easier to interpret the plot. GG23 and GG17 project low on variable U3. GG23 projects high on variable S6, and to some extent on variable S5. These findings are confirmed by referring to Table 3.15, and by the high axis predictivities in Table 3.17. GG22 projects low on variables S3, S5 and S6. Referring to Table 3.15, it is clear that although the values for both S3 and S6 are indeed below 0, the value for S5 is -3.29, which is indeed very low. Lastly, GG20 projects high on variables S3, S5 and S6. These results are again confirmed by Table 3.15. Specifically the values for S3 and S5.

From Figure 3.24b (PC1 $\times$ PC3), apart form the gasifiers already discussed, there is some indication that GG14 and GG15 need to be investigated. GG15 projects high on S4 and GG14 low on S1 and S6, and high on U1 and U4. This information was not available in the traditional PC1 $\times$ PC2 plot (Figure 3.24a). Referring to Table 3.15 it can be confirmed that GG15 has a high value for S4. GG14 has a low value for S6. Although the exact axes that caused the separation of especially GG14 were not immediately apparent, the CVA biplot did indicate the need to investigate this gasifier further. This emphasizes the importance of using the CVA biplot as a convenient and powerful visual tool, but keeping in mind that the biplot is a two-dimensional representation of a higher dimensional space.

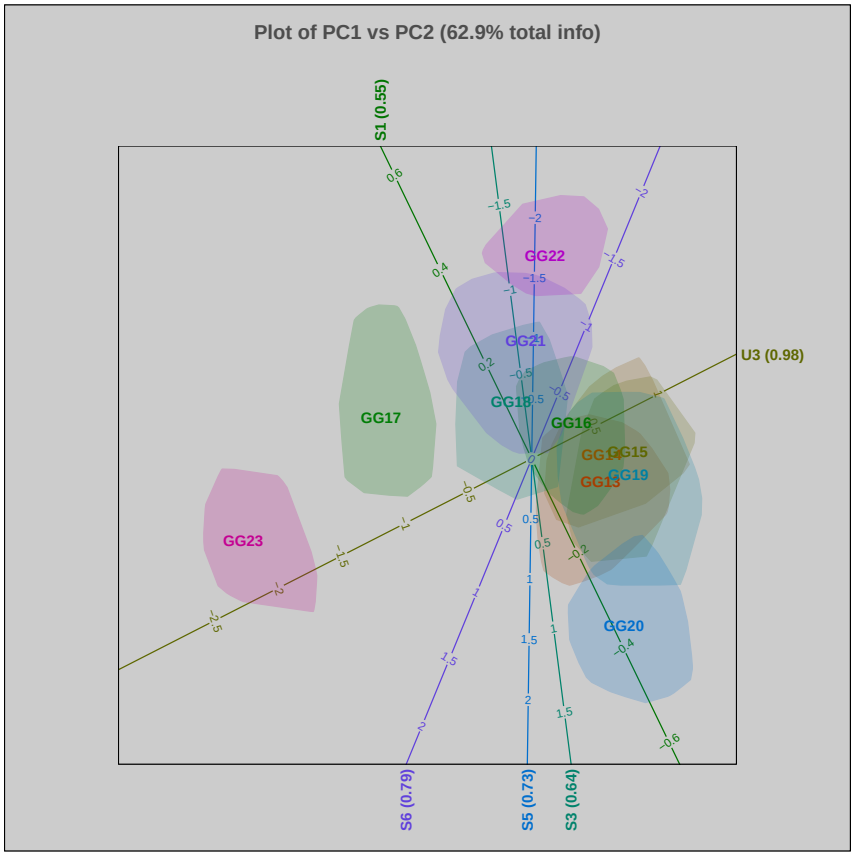
From Figure 3.24c (PC1 $\times$ PC4) it can be observed that GG16 projects low on U2 and high on S1 and S4. The axis predictivity values for these axes are however low, and it is prudent to confirm the conclusion by referring to

Table 3.15. The results are confirmed in general in that GG16 has indeed the lowest value for U2, and is slightly higher on S4.

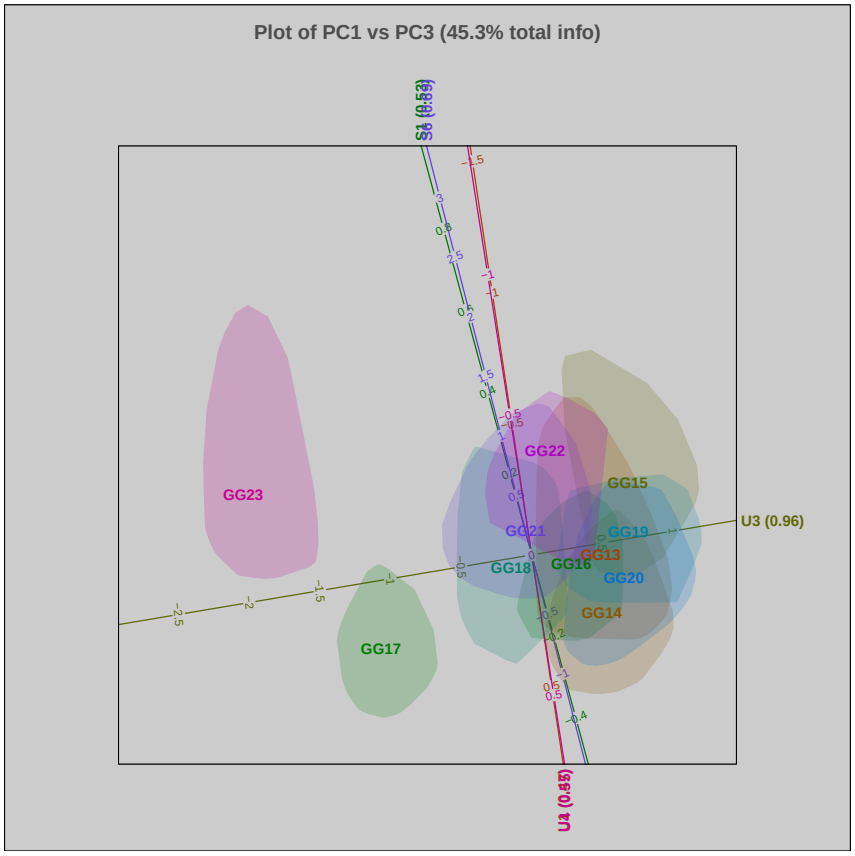
Figures 3.24d to 3.24f confirm the results obtained previously as would be expected.

Taking an overall view of Figure 3.24 it may be fair to conclude that the total number of eigenvalues to include, and therefore the combination of eigenvalues to investigate in a CVA biplot analysis, is an over simplistic view of the inherent properties of CVA (and PCA) biplots. As the biplot is built on the score plots of a combination of two principal components it is more prudent to investigate a weighted approach where, as a first step, a scree plot is generated, and thereafter the number of axes with high axis predictivities in combination with the CVA biplot quality in the original variables for the specific combination of components, are used to further exclude some of the plots and axes. If in the case study discussed an additional criterion of at least 40% variation explained in the original values was employed, Figures 3.24a to 3.24d would have been retained. These four figures contained all the information that were "lost" in Figures 3.24e and 3.24f. The suggested approach is therefore to use the scree plot to find the eigenvalues to include as a first criterion, and thereafter use the CVA biplot quality in the original variables as well as to assess the number of axes with high axis predictivities (at least two) to further filter the combinations that are investigated.

From Table 3.13 it is clear that two of the variables are not present in any of the combinations (S2 and S7). It can be concluded that these variables are not conducive to the class separation in this specific data set. It could therefore be considered to remove them from the dataset, and redo the CVA biplot analysis.

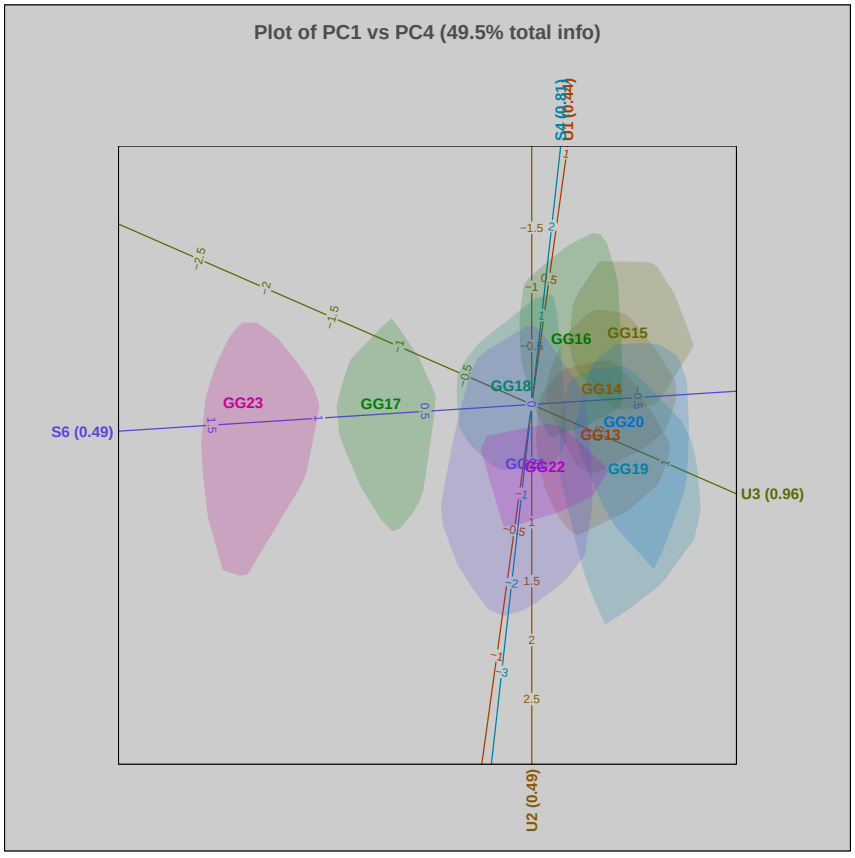


(a) CVA biplot for PC1 against PC2

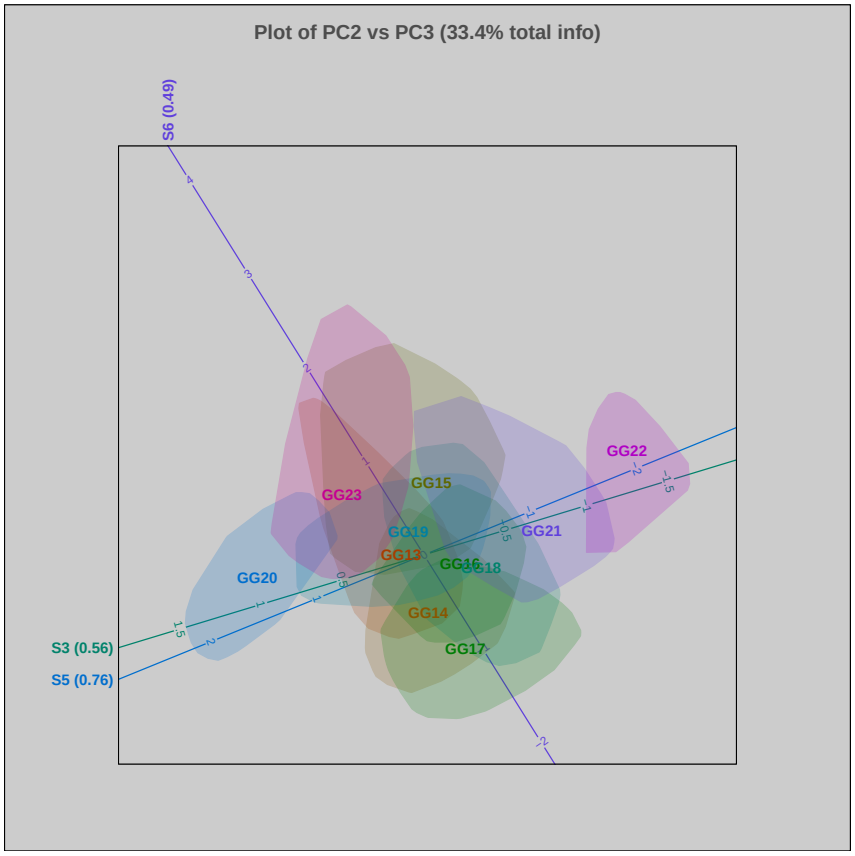


(b) CVA biplot for PC1 against PC3

Figure 3.24: CVA biplots for the different combinations of PC's

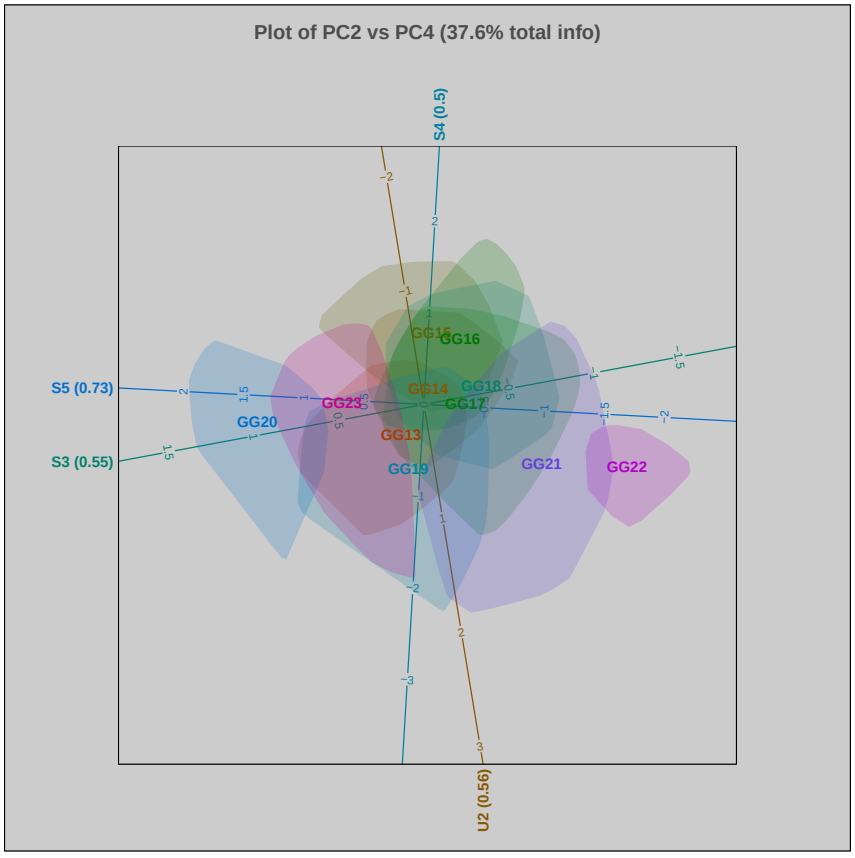


(c) CVA biplot for PC1 against PC4

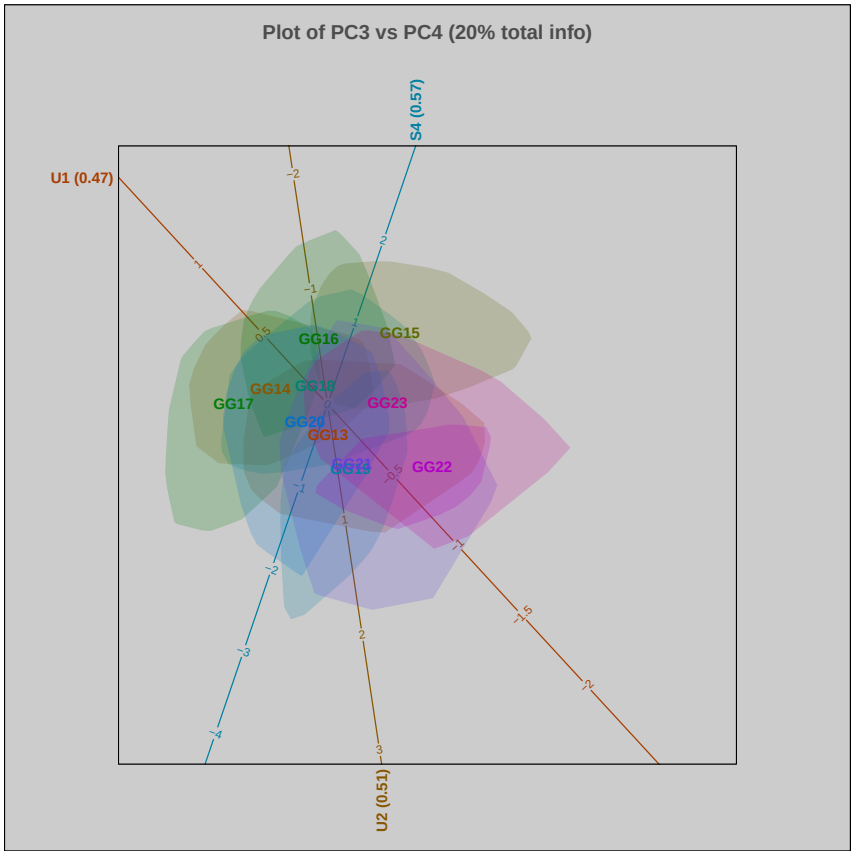


(d) CVA biplot for PC2 against PC3

Figure 3.24: CVA biplots for the different combinations of PC's continued



(e) CVA biplot for PC2 against PC4



(f) CVA biplot for PC3 against PC4

Figure 3.24: CVA biplots for the different combinations of PC's continued

3.4.5 Conclusions for CVA monitoring

In this study the CVA biplot and various measures of predictive power of the CVA biplot were investigated, and applied to an industrial case study. The first question that was addressed was the number of eigenvalues (and therefore principal components) to include in the plot. In the case study the scree plot and CVA biplot quality measures did not provide similar number of components to include. It is however recommended that, as the CVA biplots are composed of a combination of two principal components at a time, the quality in the total dimension is not the value that should be used. It is recommended that the scree plot is used to determine the number of eigenvalues (and therefore principal components) to include in the study. When considering the combinations to include the CVA biplot quality in the original variables for the specific combinations should be used in combination with the number of axes with high predictive power.

Two measures of predictive power of the biplot axes were applied to the case study. Specifically the mean standard predictive error (mspe) for PCA biplots from Alves (2012) was adapted for the CVA biplot. The mspe values for CVA biplots were contrasted with the CVA axis predictivities from Gardner-Lubbe *et al.* (2008). Very similar results were obtained in applying these two measures to the case study. It was concluded that these predictive measures are highly correlated, and any one of the two could be used.

From the case study it was concluded that more information is obtained by including the axes with high predictive power. It is therefore suggested that in any CVA biplot analysis this methodology should be applied. This is especially important for the real time on-line application of this case study. The information on the CVA biplot quality and the axis predictivities should be supplied on the plots as well to indicate to the user the confidence in the results. It is also prudent to compare the results obtained to the actual data to confirm the results. The proposed methodology is summarised in Figure 3.25.

In conclusion, CVA biplots are highly powerful visual aids in multivariate analysis. Significant information can be obtained in one glance that would be virtually impossible to obtain from tables of data. It is however necessary to exclude axes with low predictive values as to not confuse the user, as the casual user will assign equal importance to all the axes as a direct consequence of the comparison if biplots to scatter plots often made in the literature. Additionally, more than the first two eigenvalues should be investigated as there are possibly additional separations taking place on the remaining planes which are orthogonal to the first two principal components.

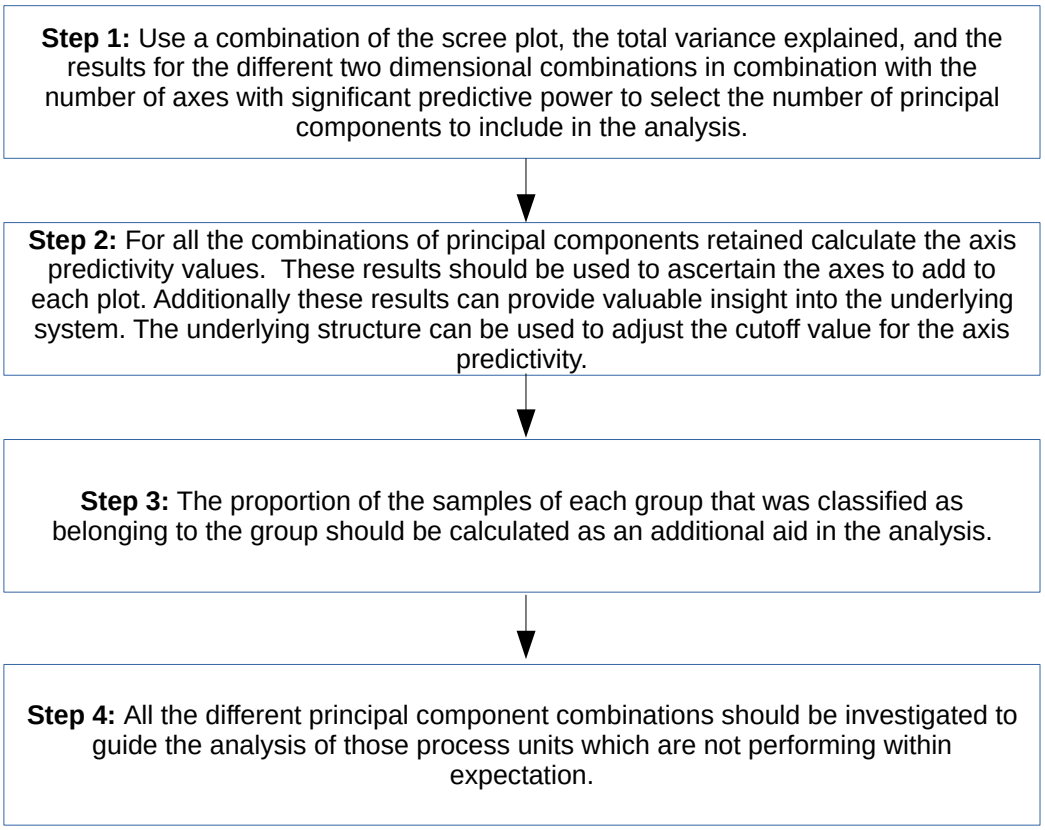


Figure 3.25: Steps for the implementation of a CVA biplot monitoring methodology

3.5 Discussion

In this chapter an empirical multivariate statistical process evaluation and a monitoring methodology were proposed. A brief summary of the methodology will now be provided. Finally, some shortcomings of a purely data driven approach will be discussed.

The first step in an empirical multivariate statistical evaluation and monitoring process is the selection of an appropriate reference data set. In Section 3.2 Generalised Orthogonal Procrustes Analysis (GOPA) was applied as a criterion for the selection of the optimal train and the optimal combination of the number of weeks as the reference set for all the production processes. PCA analyses and biplot displays were used to visualize and to interpret the results from the GOPA analysis. These interpretations provide important insight into and quantification of the relationships between the variables on the production facility. The application of the GOPA for reference set selection, and the accompanying PCA analyses are new in the multivariate process monitoring literature. It is recommended that the methodology proposed and demonstrated in this chapter for reference set selection, is applicable to any process with multiple identical production units.

The steps for determining the optimal reference set for multivariate monitoring of multiple production processes are summarized in Figure 3.11.

The results from the GOPA minimization provide a mechanism for selecting the optimal combination of time units for a specified number of units.

In Section 3.3.3 the use of the PCA biplot was proposed for short term real time process evaluation and monitoring. Specifically the methodology depicted in Figure 3.20 was proposed.

In Section 3.4.3 the use of the CVA biplot was proposed for longer term (but real time) monitoring of differences between the gasifiers on a train. The methodology as depicted in Figure 3.25 was proposed.

Following the proposed multivariate statistical evaluation and monitoring approach, the process units and periods are identified that should be investigated further in more detail. However, there are some shortcomings in this approach.

3.5.1 Shortcomings of an empirical approach

Although an empirical multivariate approach to statistical evaluation and monitoring can lead to significant insight into the underlying production process, there are some shortcomings. Most notably, there exists a large body of process knowledge in the production engineers that can enhance the understanding of the underlying process. As an example consider Figure 3.26. Clearly there exists a large variation around variables U1, U4 and S7. However, these variables are all related to reactor load and the engineer will therefore "expect" these lower values. At certain periods when gasification is not the bottleneck all the

gasifiers will run at a lower load, and this will again be expected behaviour. Therefore, although it is correct from an empirical perspective to flag these periods as "abnormal" operation, from a process perspective these values are actually inside the expected operating region. Additionally, deviation from the target or expected value is not equally important for all the variables. Some measure of scaling and weighting for importance should therefore be applied in practice.

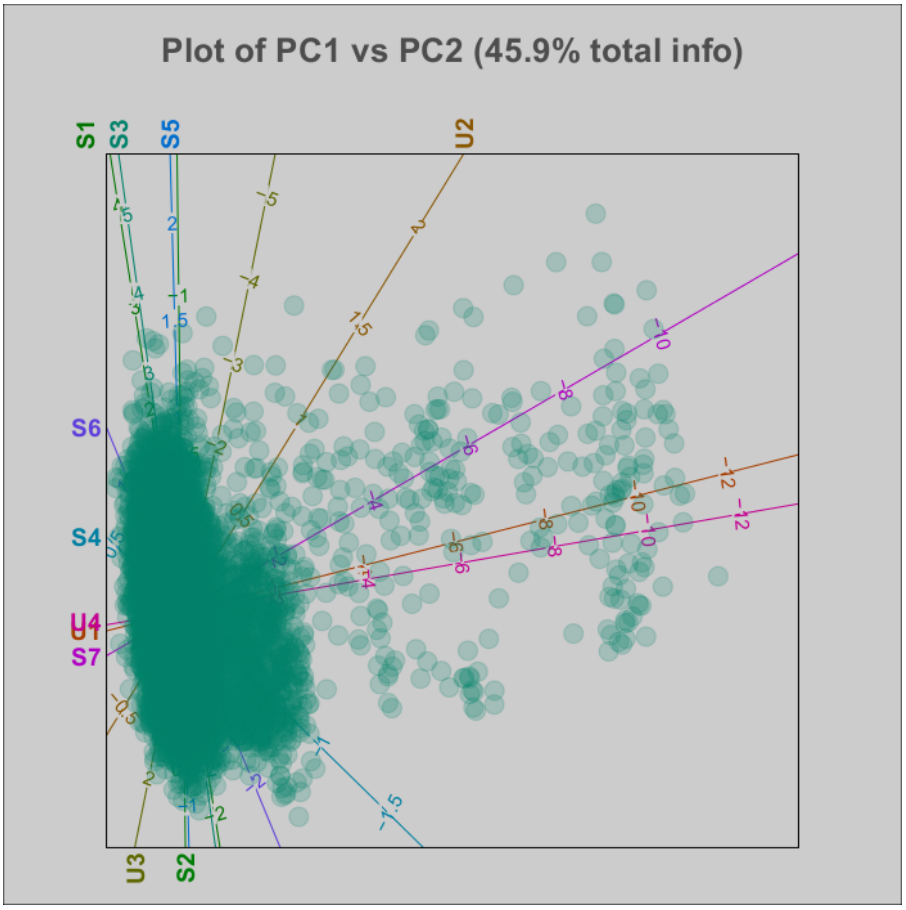


Figure 3.26: PCA Biplot of first two principal components

An additional constraint in practice is the large volume of information available. The full production facility consists of 84 gasifiers, and considering all the combinations of principal components for all 84 gasifiers for both the PCA and CVA biplots is a daunting task. The creation of a single visual display to guide the user to the gasifiers to investigate further is therefore a necessity for the practical implementation of any real time process evaluation and monitoring methodology.

In summary, in this chapter a real-time multivariate process monitoring approach for the Coal Gasification Facility was presented. This includes a

novel approach utilising Generalised Orthogonal Procrustes Analysis to find the optimal units and time period to employ as a reference set. Principal Component Analysis (PCA) and Canonical Variate Analysis (CVA) theory and biplots were evaluated and extended for the real-time monitoring of the plant.

Both the creation of an index of performance of individual gasifiers combined with the fundamental engineering knowledge, and the application thereof to this study, will be discussed and demonstrated in the next chapter.

Chapter 4

Gasifier Performance Index

In Chapter 3 a multivariate empirical approach to process monitoring was developed and proposed. Some shortcomings were highlighted in Section 3.5.1 that will be addressed in this chapter. Specifically, in Section 4.1 a fundamental index will be reviewed for gasifier performance referred to as the fundamental GPI. In contrast to this fundamental GPI a purely empirical approach is considered in Section 4.2 referred to as the empirical GPI. Finally, in Section 4.3 a combined fundamental and empirical approach will be developed and discussed referred to as the integrated GPI.

4.1 Fundamental Gasifier Performance Index (GPI)

4.1.1 Introduction

Wuzyk and Koper (1992) proposed a performance index for gasification called the Gasifier Index (GIX). In addition, they proposed a dynamic version that incorporated the gasifier load set-point value (DGIX). As part of the present study the DGIX was revisited and updated to incorporate more recent operation philosophy. The updated version is referred to as the Gasifier Performance Index (GPI). More specifically in the current study this version of the GPI will be referred to as the fundamental GPI to distinguish between the three different approaches to a gasifier index.

4.1.2 Fundamental GPI definition

The variables included in the GPI are the same as the variables discussed in Chapter 3 for process monitoring. In addition, in Table 4.1 a variable number is specified for each of the generic variable names. The variable numbers range from $1, \cdots, p$ where $p = 11$. The GPI value is defined as a weighted offset from

a recommended value for each variable. Due to confidentiality constraints these actual recommended values cannot be provided here.

Table 4.1: Variable types

Number	Variable	Type
1	U1	Utility
2	U2	Utility
3	U3	Utility
4	S1	Stability
5	S2	Stability
6	S3	Stability
7	S4	Stability
8	S5	Stability
9	S6	Stability
10	S7	Stability
11	U4	Utility

For each variable $i = 1, \cdots, p$ define:

- Opt_i : The recommended value for variable i.
- $Dmin_i$: Delta value for the minimum value for variable i i.e., $min_i = Opt_i - Dmin_i$.
- $Dmax_i$: Delta value for the maximum value for variable i i.e., $max_i = Opt_i + Dmax_i$.
- $AdjF_i$: Adjustment factor for variable i. The adjustment calculation will be discussed in detail below.
- Adj_i : Adjustment indicator for variable i i.e., should variable i be adjusted.
- w_i : Weight assigned to variable i in the GPI calculation.
- x_{ti} : Actual value for variable i at time t.
- $Fmin_i$: Absolute lower bound for variable i.
- $Fmax_i$: Absolute upper bound for variable i.
- v_{ti} : Flag to indicate if the actual value of variable i at time t (x_{ti}) is in the range $Fmin_i \leq x_{ti} \leq Fmax_i$.

Additionally m is defined to be the overall factory load variable, and m_{opt} to be the optimal set-point.

The dynamic corrections for the variables are defined as follows:

- A correction factor ($Corf$) is calculated as the ratio of the difference between the current factory load set-point m and the optimal factory load set-point (4.1.1).

$$Corf = \frac{m - m_{opt}}{m_{opt}} \tag{4.1.1}$$

- All the gasification variables are not adjusted with this factor. The adjustment factor $AdjF_i$ is an additional factor applied to the correction factor, and the indicator variable (Adj_i) indicates if variable i should be adjusted with the correction factor.

- The calculations for the adjustments for the recommended operating limits and optimum values ($Dmin_i$, $Dmax_i$, and Opt_i) are provided below ((4.1.2) - (4.1.4)).

$$Dmin_i^* = Dmin_i \times (1 + Corf \times AdjF_i \times Adj_i) \tag{4.1.2}$$
$$Opt_i^* = Opt_i \times (1 + Corf \times AdjF_i \times Adj_i) \tag{4.1.3}$$
$$Dmax_i^* = Dmax_i \times (1 + Corf \times AdjF_i \times Adj_i) \tag{4.1.4}$$

The recommended Opt_i values for variables U1, U3 and U4 are calculated as follows:

$$Opt_{U4}^* = \frac{m}{100} \times c \tag{4.1.5}$$

where c is a constant value defined in the plant operating procedure.

$$Opt_{U1}^* = \frac{Opt_{U4}^*}{Opt_{U2}^*} \tag{4.1.6}$$

$$Opt_{U3}^* = Opt_{U4}^* \times 0.175 \tag{4.1.7}$$

The variable v_{ti} is defined as

$$v_{ti} = \begin{cases} 1, & \text{if } Fmin_i \leq x_{ti} \leq Fmax_i \\ 0, & \text{otherwise} \end{cases} \tag{4.1.8}$$

After adjusting for the correction factor the GPI at time t is calculated using (4.1.9) where x_{ti} is the actual value for the i-th variable in Table 4.1 at time t. The `ifelse` statement is interpreted as: if x_{ti} is less than Opt_i^* divide by $Dmin_i^*$, else divide by $Dmax_i^*$. The floor function map a real number to the largest previous integer.

$$GPI_t = \text{floor}\left[100 \times \frac{\sum_{i=1}^{p}\left[\left(\frac{|Opt_i^* - x_{ti}|}{\text{ifelse}[[Opt_i^* - x_{ti}]<0, Dmax_i^*, Dmin_i^*]}\right) \times w_i v_{ti}\right]}{\sum_{i=1}^{p} w_i v_{ti}}\right] \tag{4.1.9}$$

Therefore, the GPI is a weighted sum of the variables, adjusted by the correction factors and scaled to the same range. Also if x_{ti} is outside of the range $Fmin_i \geq x_{ti} \leq Fmax_i$, the weight is multiplied by 0 and variable i will have 0 contribution to the GPI_t. Note that GPI_t is undefined if all $v_{ti} = 0$ for $i = 1, \cdots, p$. Furthermore, note the GPI_t value can be greater than 100.

Finally, (4.1.10) was derived to calculate a standardized variable contribution to the GPI_t value.

$$GPIContr_{ti} = \frac{\left[\left(\frac{|Opt_i^* - x_{ti}|}{\text{ifelse}[[Opt_i^* - x_{ti}]<0, Dmax_i^*, Dmin_i^*]}\right) \times w_i v_{ti}\right]}{\sum_{j=1}^{p}\left[\left(\frac{|Opt_j^* - x_{tj}|}{\text{if}[[Opt_j^* - x_{tj}]<0, Dmax_j^*, Dmin_j^*]}\right) \times w_j v_{ti}\right]} \times 100 \quad (4.1.10)$$

4.1.3 Fundamental GPI implementation

The GPI value (4.1.9) is calculated on 15 minute time weighted average data for all the variables in Table 4.1 for a 24 hour period, and displayed on a custom heatmap. Refer to Figure 4.1, which shows the GPI graph for the Eastern Factory. Each cell in Figure 4.1 contains the GPI graph for the specific gasifier. The colour of the cell is derived from the average GPI value over the 24 hour period and ranges from light blue (within expectation) to dark red (extreme deviation). Referring to the graph in Figure 4.2 for GG26, the dashed line depicts the average GPI value over the 24 hour period. The solid line depicts the real time GPI value over the 24 hour period. The scale of the GPI graphs is 0-120, and values higher than 120 are capped at 120. Gaps in the solid line, for example on the graph for GG15 indicate that the gasifier went offline. Empty white blocks indicate that the gasifier were offline for the entire 24 hour period.

Referring to Figure 4.2, the background color corresponds to a GPI value of between 70 and 80 over the most recent 24 hours. The GPI is in essence a weighted deviation from the target values; therefore GPI values closer to 0 are better (indicates operating within expectation). The variable contribution plot as shown in Figure 4.3 is calculated with (4.1.10) for the last 15 minutes. From Figure 4.3 it can be concluded that variable S3 contributed the most to the calculated GPI value. The trend plots for GG26 are shown in Figure 4.4 for the last 24 hours. Note that the x and y axes were removed from these plots due to confidentiality concerns. In addition to the actual trend, the dynamic recommended optimum value (Opt_i in (4.1.3)), minimum value ($Opt_i - Dmin_i$) and maximum value ($Opt_i + Dmax_i$) are also displayed on the graphs. It is clear that S3 was above the maximum value for the period, and this would explain the high contribution to the GPI. Figure 4.4j depicts the specific gasifier load variable, and it is of interest to note that for a specific period in the middle of the 24 hours the gasifier load was reduced to be lower than the recommended value. Referring to Figures 4.4a and 4.4k it can be

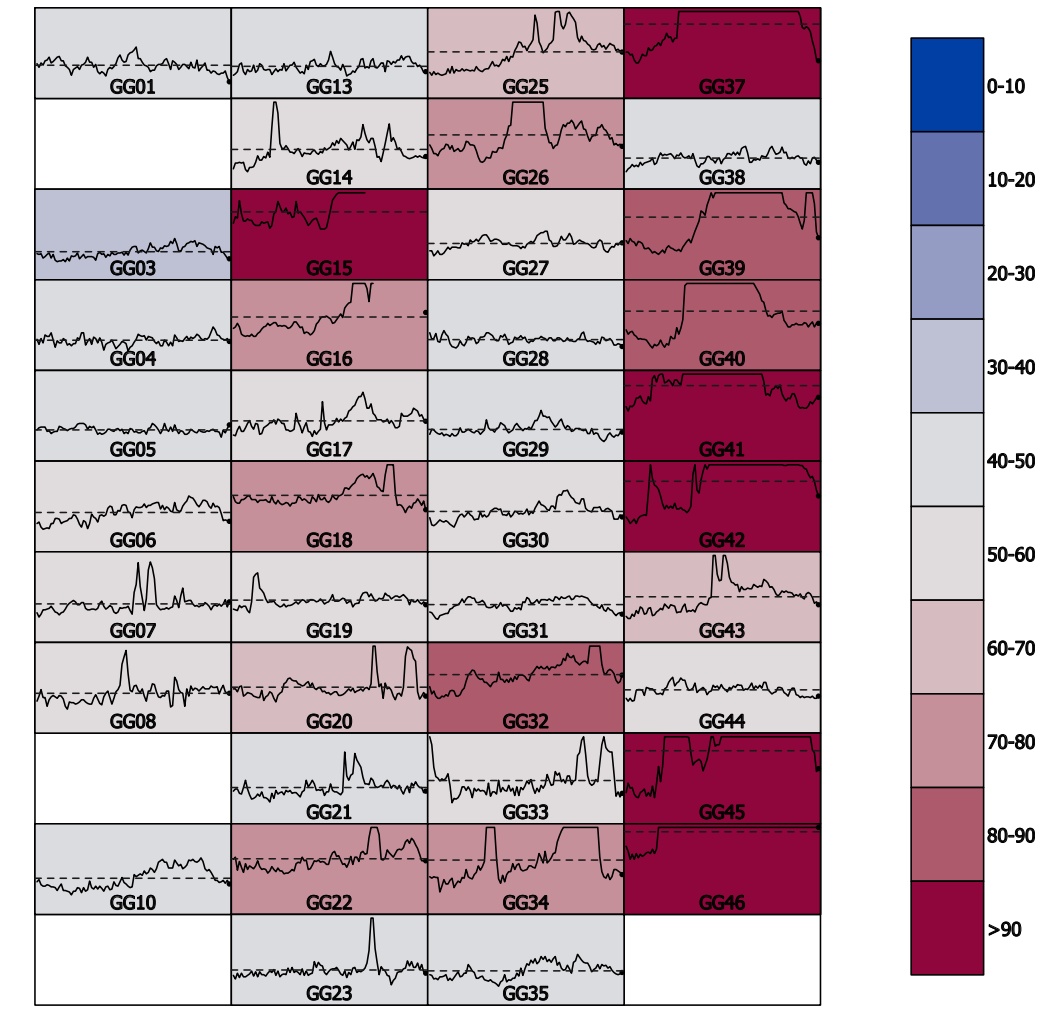


Figure 4.1: Fundamental GPI graph

Figure 4.2: Zoomed in mini graph for GG26

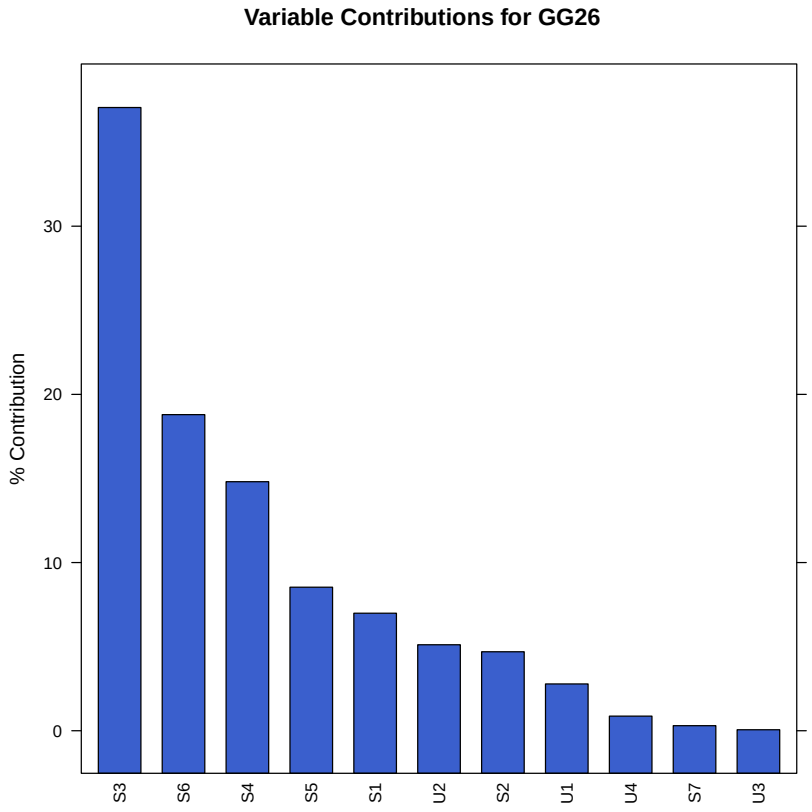


Figure 4.3: Variable contribution graph for GG26

observed that both U1 and U4 closely followed the recommended value except for the period that the gasifier specific load variable was lower. This is to a lesser extend true for U3. Variables U1, U3 and U4 are closely linked to the factory load variable from equations (4.1.5) - (4.1.7). The gasifier specific load variable however overrides the factory load variable. This can normally be attributed to a gasifier trip or cutback. For GG26 a spike can be observed in S1 prior to the load cutback, and it can therefore be concluded that the gasifier was cut back due to a high S1 value.

The GPI graph contains a huge amount of information in one easy to interpret visual display. For example, from Figure 4.1 it is observed that train four is experiencing some issues as the GPI values are generally higher than the other trains, and should be investigated further to find the source of the deviation. In contrast, train one is performing very stable for this specific period.

4.1.4 Summary for the fundamental GPI

The fundamental GPI has several advantageous characteristics. Some of the inherent advantages are:

1. There is no need to define a reference set.
2. The GPI calculation is computationally efficient.
3. The contribution of each variable to the overall GPI is easy to calculate.
4. The GPI calculation is intuitive and easy to relate to the actual production process. This leads to ease of acceptance by the production engineers.

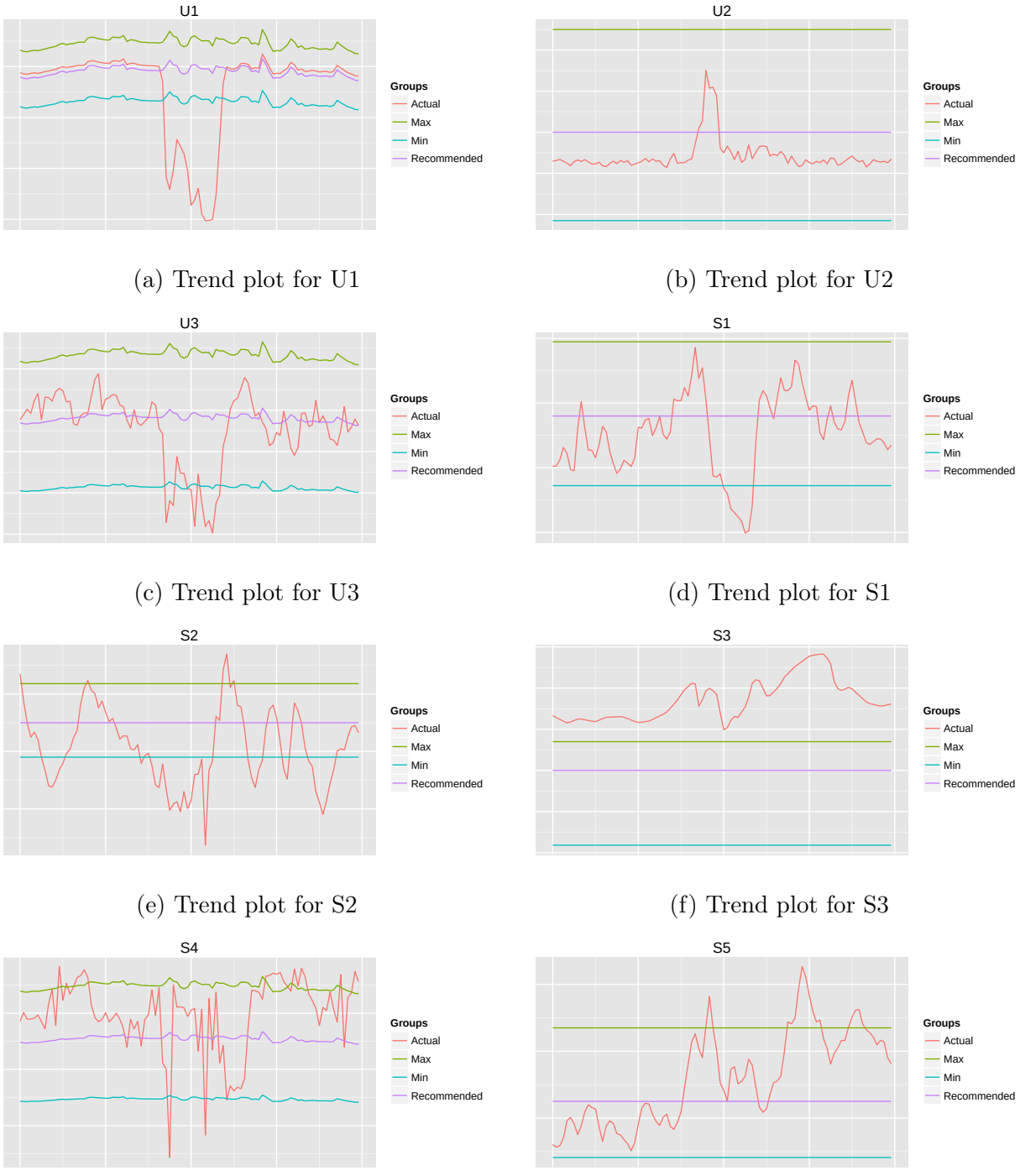
The fundamental approach however possesses two big disadvantages:

1. The GPI value is subjective as all the underlying input values are supplied by subject matter experts.
2. The GPI is by definition univariate, and does not take into account any multivariate relationships between the variables.

In addition to providing the recommended values the subject matter experts also provide the delta values for the minimum and maximum, as well as the weighting for each variable. The GPI is in fact double weighted as both the actual weight and the delta values will affect the overall impact of the variable on the GPI. The impact of the weight is intuitive, but the impact of the delta values is more subtle.

Figure 4.5 demonstrates the effect of different delta values on the shape of the GPI contribution by variable i. It can be observed that these shapes are an implementation of desirability functions as discussed in Kim and Lin (1998, 2000) and Coetzer *et al.* (2008). A steeper slope will have a similar effect as a higher weight. The slope of the desirability function may be different for approaching the optimum (target) from the left (from below), compared to approaching it from the right (from above). The slope of the function determines the importance (or strictness) for each deviation from the target. Therefore the delta values will have a direct impact on not only the overall weighting of the variable, but also on the detection rate for deviation from the target. Desirability functions are powerful tools, and although outside the scope of this study, different types of desirability functions can be considered and compared for the fundamental GPI function (see Coetzer *et al.* (2008)).

The fundamental approach can be generalised to any process given that the required process knowledge is available. In the next section a purely data driven approach to the GPI will be developed and demonstrated.



(a) Trend plot for U1

(b) Trend plot for U2

(c) Trend plot for U3

(d) Trend plot for S1

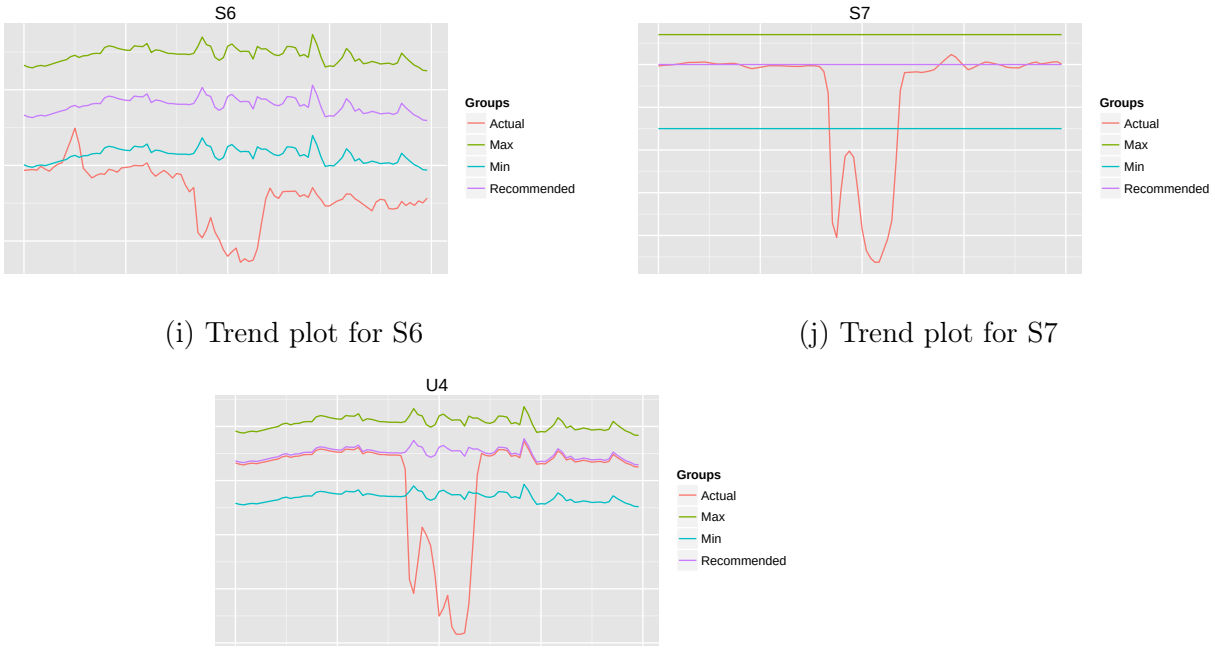
(e) Trend plot for S2

(f) Trend plot for S3

(g) Trend plot for S4

(h) Trend plot for S5

Figure 4.4: Trend plots of the process variables for GG26



(i) Trend plot for S6

(j) Trend plot for S7

(k) Trend plot for U4

Figure 4.4: Trend plots of the process variables for GG26 continued

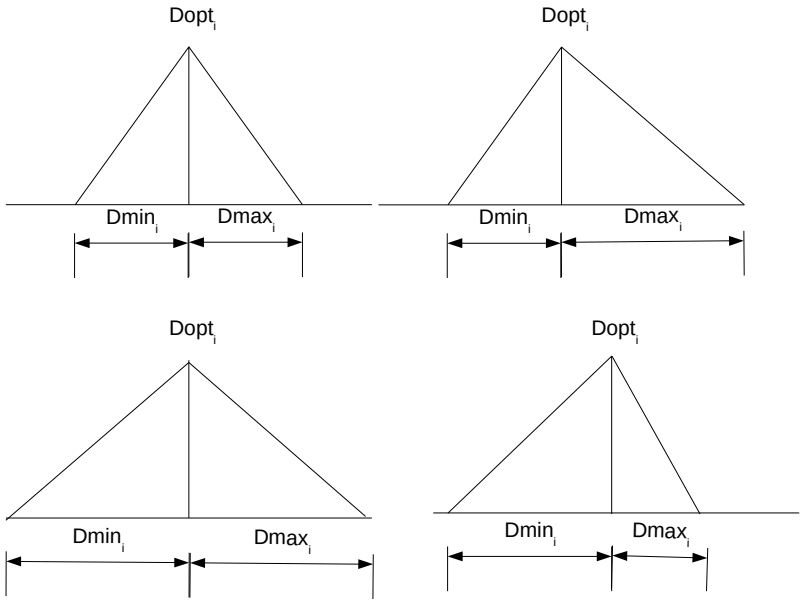


Figure 4.5: Demonstration of the effect of different delta value choices for the minimum and maximum

4.2 Empirical Gasifier Performance Index

4.2.1 Introduction

The topic of multivariate process control has received much attention in literature (Cox, 2001; Ferrer, 2014; Kourti and MacGregor, 1995; MacGregor, 1997; MacGregor and Kourti, 1995; Qin, 2014; Russell *et al.*, 2000; Yeh *et al.*, 2006). The objective of statistical process control (SPC) is to monitor the performance of a process over time to verify that it is remaining in a "state of statistical control". Such a state of control is said to exist if certain processes or product variables remain close to their desired values and the only source of variation is 'common-cause' variation, that is variation which affects the process all the time and is essentially unavoidable (Kourti and MacGregor, 1995).

Traditionally, SPC charts (Shewart, CUSUM and EWMA) are used to monitor a small number of key product variables ($\boldsymbol{Y}$) in order to detect the occurrence of any event having a "special" or "assignable" cause. By finding assignable causes, long term improvement in the process and product quality can be achieved by eliminating the causes or improving the process or its operating procedures. However, monitoring only a few quality variables is inadequate for most modern process industries. The traditional SPC approaches ignore the fact that with computers hooked up to nearly every process, massive amounts of data are collected routinely every few seconds on many process variables ($\boldsymbol{X}$), such as temperatures, pressures, flow rates etc (Ferrer, 2014). Final product quality variables such as polymer properties, gasoline octane numbers etc., are available on a much less frequent basis, usually from off-line laboratory analyses. All such data should be used to extract information in an effective scheme for monitoring and diagnosing operating performance (Kourti and MacGregor, 1995). These data are of limited value without appropriate processing, especially with respect to the discovery of abnormal events derived from the interaction between variables, tracking of process drift, and so forth (Aldrich *et al.*, 2004).

However, process variables are not independent of one another. Only a few underlying events are driving a process at any time, and all these measurements are simply different reflections of the same underlying events (MacGregor and Kourti, 1995). Therefore, performing one variable at a time analysis as though the events are independent, makes interpretation and diagnosis very difficult and inefficient. Such methods only look at the magnitude of the deviation in each variable independently of all others. Only multivariate methods that treat all the data simultaneously can in addition extract information on the directionality of the process variations, that is on how all the variables are behaving relative to one another (Kourti and MacGregor, 1995). Multivariate Statistical Process Control (MSPC) is increasingly being recognized as a valuable tool for providing early warnings of process changes, the identification of

potential plant faults, process malfunctions and process disturbances, and for enabling a deeper understanding of the process to be achieved (Weighell *et al.*, 2001).

When important events occur in processes they may be difficult to detect because the signal to noise ratio can be very low in each variable. However, multivariate methods can extract meaningful information from observations on many variables and can reduce the noise levels through averaging (Kourti and MacGregor, 1995).

4.2.2 Empirical GPI definition

In the current study the focus is on applying multivariate statistics and biplot related theory to the problem of multivariate process monitoring. In the definition of a data driven (empirical) performance index for the gasifiers, an important criterion is that the results should relate to the work discussed in Section 3.3 for process monitoring biplots. Specifically, (3.3.19) in Section 3.3.3 is directly related to the well known T^2-statistic.

The T^2-statistic is derived as follows (MacGregor and Kourti, 1995; Russell *et al.*, 2000):

- Given an $n \times p$ column-wise mean centered and unit variance reference set $\boldsymbol{X}$ with the p variables as the columns and the n samples as the rows, the sample covariance matrix is defined as

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X} \tag{4.2.1}$$

- The eigenvalue decomposition of the covariance matrix $\boldsymbol{S}$

$$\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T \tag{4.2.2}$$

  has the following properties:

  - $\boldsymbol{V}$ is orthogonal i.e., $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$.
  - The projection $\boldsymbol{y} = \boldsymbol{V}\boldsymbol{x}$ of a new observation vector $\boldsymbol{x}$ transforms $\boldsymbol{x}$ into a set of uncorrelated variables (scores).
  - The variance of the i-th element of $\boldsymbol{y}$ is equal to the i-th value on the diagonal of $\boldsymbol{\Lambda}$ (the eigenvalues).

- The T^2-statistic for vector $\boldsymbol{x}$ is then given by

$$T^2 = \boldsymbol{x}^T\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^T\boldsymbol{x} \tag{4.2.3}$$

Note that identical results can be obtained by performing a singular value decomposition on

$$\frac{1}{\sqrt{n-1}}\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{4.2.4}$$

where $\boldsymbol{V}$ is identical to the $\boldsymbol{V}$ in (4.2.2) and $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$. $\boldsymbol{\Sigma}$ contains the singular values on the diagonal. The T^2-statistic can then be defined as

$$T^2 = \boldsymbol{x}^T\boldsymbol{V}(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma})^{-1}\boldsymbol{V}^T\boldsymbol{x} \tag{4.2.5}$$

Although (4.2.2) and (4.2.4) lead to identical results from an algebraic perspective, from biplot theory (4.2.2) yields a monoplot and (4.2.4) yields a biplot (Gower *et al.*, 2011). PCA biplots are constructed by plotting the rows of $(\boldsymbol{U\Sigma})_{r^*}$ representing the samples, and the rows of $\boldsymbol{V}_{r^*}$ giving the direction for the variables (see Section 3.3.1 for more detail). Both the samples and variables are therefore represented on a single plot simultaneously, hence the "bi" in the name biplot refers to the two types of entities exhibited simultaneously on the plot.

Gower *et al.* (2011) defines a monoplot as a display where only one type of entity is displayed in two or more dimensions. For example, (4.2.2) will lead to a covariance monoplot when the rows of $(\boldsymbol{V}\sqrt{\boldsymbol{\Lambda}})_{r^*}$ are plotted, where $(\boldsymbol{V}\sqrt{\boldsymbol{\Lambda}})_{r^*}$ refers to the first r^* rows of $\boldsymbol{V}\sqrt{\boldsymbol{\Lambda}}$. The inner product $((\boldsymbol{V}\sqrt{\boldsymbol{\Lambda}})_{r^*})((\boldsymbol{V}\sqrt{\boldsymbol{\Lambda}})_{r^*})^T = \boldsymbol{V}_{r^*}\boldsymbol{\Lambda}_{r^*}\boldsymbol{V}_{r^*}^T$ approximates the covariance matrix $\boldsymbol{S}$. The inner product is therefore found from pairs of points of the same kind, both representing the variables (Gower *et al.*, 2011).

If the actual covariance matrix of the in control data is known then the T^2-statistic follows a χ^2 distribution with p (the number of variables) degrees of freedom. When the actual covariance matrix for the in control data is not known but estimated by the covariance matrix of the reference set (4.2.1), the threshold for the T^2-statistic at significance level α is given by

$$T_\alpha^2 = \frac{p(n-1)(n+1)}{n(n-p)}F_\alpha(p, n-p) \tag{4.2.6}$$

where $F_\alpha(p, n-p)$ is the upper $100\alpha\%$ critical point for the F-distribution with p and $n-p$ degrees of freedom (Russell *et al.*, 2000).

Although it is possible to use the T^2-statistic defined over the full dimensional space, there is a risk that small errors in the loading vector corresponding to the smaller singular values can have a big impact on the statistic as the square of the singular values are inverted in (4.2.5). The smaller singular values are also prone to errors as it contains a small signal to noise ratio. It is therefore beneficial to calculate the T^2-statistic only on the larger singular values (Ferrer, 2014). The number of eigenvalues to retain was discussed in detail in Section 3.3.4, and the results are directly applicable to the T^2-statistic. If the number of dimensions is defined as r^*, similar to the discussion in Section 3.3.2, $\boldsymbol{V}_{r^*}$ is defined as the first r^* columns of $\boldsymbol{V}$ and $\boldsymbol{\Sigma}_{r^*}$ as the first r^* rows and columns of $\boldsymbol{\Sigma}$. The lower dimensional T^2-statistic for a new observational vector $\boldsymbol{x}$ is then defined as

$$T^2 = \boldsymbol{x}^T\boldsymbol{V}_{r^*}(\boldsymbol{\Sigma}_{r^*}^T\boldsymbol{\Sigma}_{r^*})^{-1}\boldsymbol{V}_{r^*}^T\boldsymbol{x} \tag{4.2.7}$$

and subsequently the threshold for the T^2-statistic at significance level α is given by

$$T_\alpha^2 = \frac{r^*(n-1)(n+1)}{n(n-r^*)} F_\alpha(r^*, n-r^*) \tag{4.2.8}$$

If dF is defined as the F-distribution function and q as

$$q = T^2 \frac{n(n-r^*)}{r^*(n-1)(n+1)} \tag{4.2.9}$$

then the α (confidence) value for a specific T^2-value can be computed as follows:

$$\alpha = dF(q, r^*, n-r^*) \tag{4.2.10}$$

Russell *et al.* (2000) page 43 defines an approach to the calculation of the contribution of each variable to the overall T^2-value as follows:

1. Calculate the normalised scores for all r^* scores

$$\left(\frac{t_i}{\sigma_i}\right)^2 \tag{4.2.11}$$

   Determine the a scores responsible for the out of control status by comparing each $\left(\frac{t_i}{\sigma_i}\right)^2$ to $(T_\alpha^2)^{\frac{1}{r^*}}$ and only retain the scores where

$$\left(\frac{t_i}{\sigma_i}\right)^2 > \left(T_\alpha^2\right)^{\frac{1}{r^*}} \tag{4.2.12}$$

2. Calculate the variable contribution for each variable x_j to the out of control scores t_i

$$c_{ij} = \frac{t_i}{\sigma_i^2} v_{ij}(x_j - \mu_j) \tag{4.2.13}$$

   where v_{ij} is the (i,j)-th element of $\boldsymbol{V}$ and c_{ij} is the (i,j)-th element of an $r^* \times p$ matrix $\boldsymbol{C}$ with default values 0.

3. Now, set any $c_{ij} < 0$ equal to 0.

4. Calculate the total variable contribution for the j-th variable x_j

$$tc_j = \sum_{i=1}^{r^*} c_{ij} \tag{4.2.14}$$

   where $\boldsymbol{tc}$ is a vector of length p.

4.2.3 Empirical GPI implementation

To achieve functional parity with the fundamental GPI the empirical GPI needs to provide a single index GPI_t for each time t over all p variables. Additionally

- The index should be easily interpretable.
- The range should be similar to the fundamental GPI (this is a usability constraint as the users should be able to use the prior knowledge of the fundamental GPI to interpret the empirical GPI).
- The range should be independent of the number of variables p and the number of dimensions r^*.

Recall from Chapter 3 that the reference data set ($\boldsymbol{X}$) consists of the data for Train 1 from M20 as specified in Section 3.2.4.1. The selection of the number of principal components to include for the reference set is discussed in detail in Section 3.3.4.1 where it was concluded that four principal components are to be retained. The scree plot for the reference set is depicted in Figure 4.6.

The T^2-statistic over the same time period used in Section 4.1.3 for GG26 is depicted in Figure 4.7. Comparing Figure 4.7 with Figure 4.2, it is observed that although the peak in the GPI in Figure 4.2 is magnified between indices 40 and 60, the remaining values are very small due to the large range of the T^2-values. Interpretation of the T^2 value is also not easy without the necessary statistical background. Generally, in the literature, a confidence value i.e., $\alpha = 90\%$, is added to the plot to indicate which values are out of expected performance. However, in this study an index value comparable to the fundamental GPI is required.

A more descriptive index value is the confidence value (4.2.10) at each T^2-value. Equation (4.2.10) can be used to calculate this value from the T^2-values. The proposed index can be interpreted as the probability that the specified T^2-value is greater than or equal to the T^2-value of the reference set. The calculated index values are depicted in Figure 4.8 for GG26. Comparing Figure 4.8 to Figure 4.2 similar trends are observed. Furthermore, these index values have the following properties:

- The index values are directly interpretable as the probability that the specified T^2-value is greater than or equal to the T^2-value of the reference set.
- The index yields a range between 0 and 100, where 0 is closer to the mean of the reference set, and 100 is furthest away from the mean of the reference set.

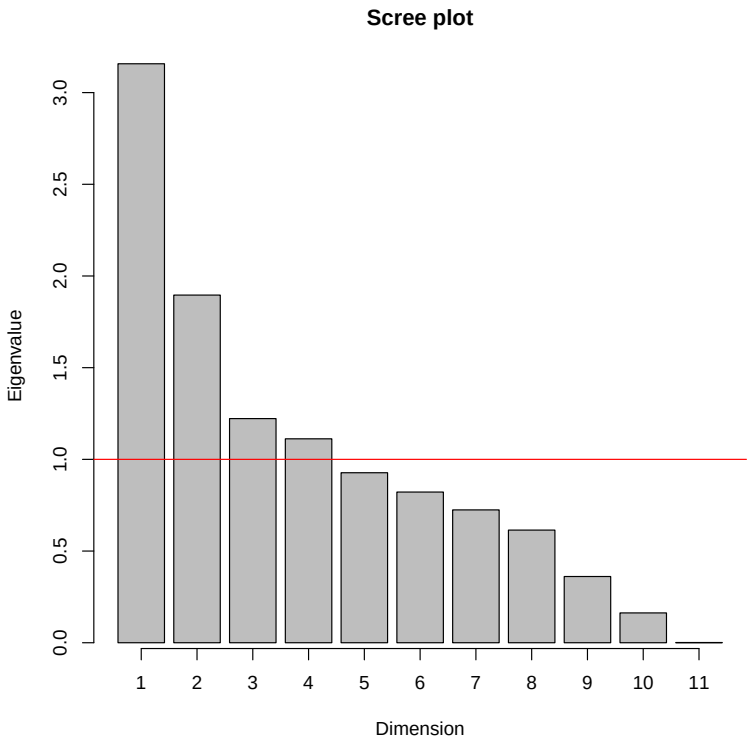


Figure 4.6: Scree Plot for reference set ($\boldsymbol{X}$) for Eastern factory

- The range of the index values is independent of the number of dimensions, and the number of variables.

Therefore, (4.2.10) is proposed and recommended as an empirical GPI. No evidence was found in the statistical and engineering literature where the confidence value have been applied as and index for multivariate process monitoring.

Figure 4.9 depicts a heatmap of the empirical GPI similar to the fundamental GPI in Figure 4.1. Comparing Figure 4.1 and 4.9 it is observed that the same gasifiers are highlighted for deviations from expected performance.

As discussed in Section 3.3.3, PCA biplots provide for both the flagging of unexpected behaviour as well as the contributing variables on one graph. However, investigating all the principal component combinations can be a time consuming process, and the T^2 contribution value from (4.2.14) can be used in combination with the biplots to guide the user. In addition, the score contribution in (4.2.12) can be used to flag the principal component combinations that contributed most to the out of control T^2-value. Therefore, only the relevant principal component combinations can be displayed which will aid in the interpretation of the results. The application of score contributions to the T^2-value and the selection of biplot principal component combinations to display is a novel approach to multivariate process monitoring.

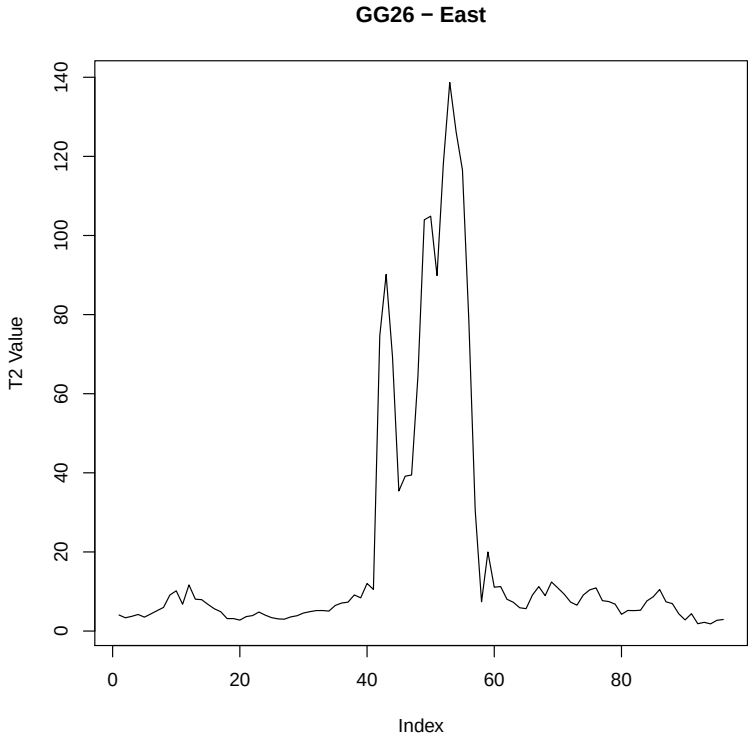


Figure 4.7: T^2-Statistic for GG26 over time

As the GPI is calculated over a 24 hour period using 15 minute aggregated data, (4.2.12) was applied to each 15 minute aggregated vector ($\boldsymbol{x}_t$) and the principal components were captured having contributions to the overall T^2-value above the threshold value $((T^2_\alpha)^{\frac{1}{r^*}})$ for time t. In this specific time period for GG26 each of the four principal components contributed significantly to the T^2-value at least once, and therefore all four principal component combinations were retained for the biplot analysis. Figure 4.10 depicts monitoring PCA biplots for all the principal component combinations of the four principal components for GG26 for the specific period under investigation. Only the relevant axes with predictivity values greater than 0.35 are displayed (See discussion in Section 3.3.4.2). All the biplots including PC1 (Figures 4.10a to 4.10c) highlight the period of low gasifier load as discussed in Section 4.1.3. In addition, Figure 4.10c indicates relatively high degree of variability for variables S2 and S4. Figure 4.10d highlights the high values for variable S3. Figures 4.10d and 4.10f again indicate high degree of variability for variables S2 and S4.

The variable contributions as well as score contributions are depicted in Figure 4.11 for specific $\boldsymbol{x}_t$ values. These values will now be discussed in more detail to compare the results from the GPI index, PCA biplots and the contribution plots.

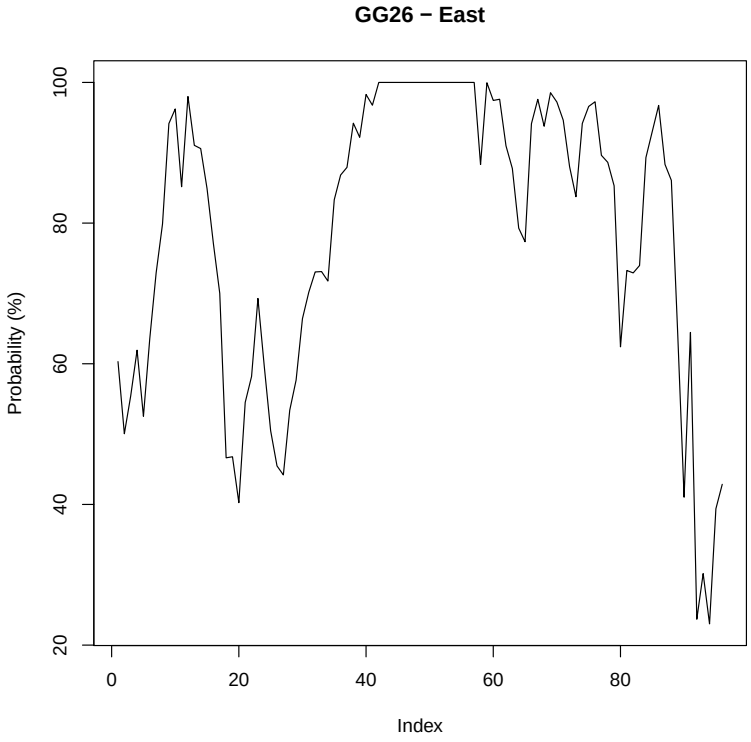


Figure 4.8: F-distribution probability values (α confidence on index) for the T^2-values for GG26

- The sharp increase in the empirical GPI index occurs at time $t = 41$. The variable and score contribution plots for $t = 41$ are depicted in Figures 4.11a and 4.11b. Principal component 2 is the only principal component that significantly contributes to the T^2-value at $t = 41$. In addition, variables S3, S1 and U3 are the three largest contributing variables. From Table 3.9 it is clear that these three variables have the highest loading values for principal component 2. Figure 4.10a also confirms that a deviation on the second principal component occurred at time $t = 41$.

- The largest empirical GPI index value occurs at time $t = 43$. From (4.2.12) principal components 1, 2, and 3 contribute significantly to the T^2-value. Figure 4.11d confirms these results, but highlights the proportionally larger effect of principal component 1. From the discussion of Table 3.9 in Section 3.3.4.3 it was concluded that principal component 1 represents the reactor load settings and therefore the variables S7, U1 and U4. The variable contribution plot in Figure 4.11c confirms these results. Figure 4.10a highlights the combined effect of principal component 1 and principal component 2 at time $t = 43$.

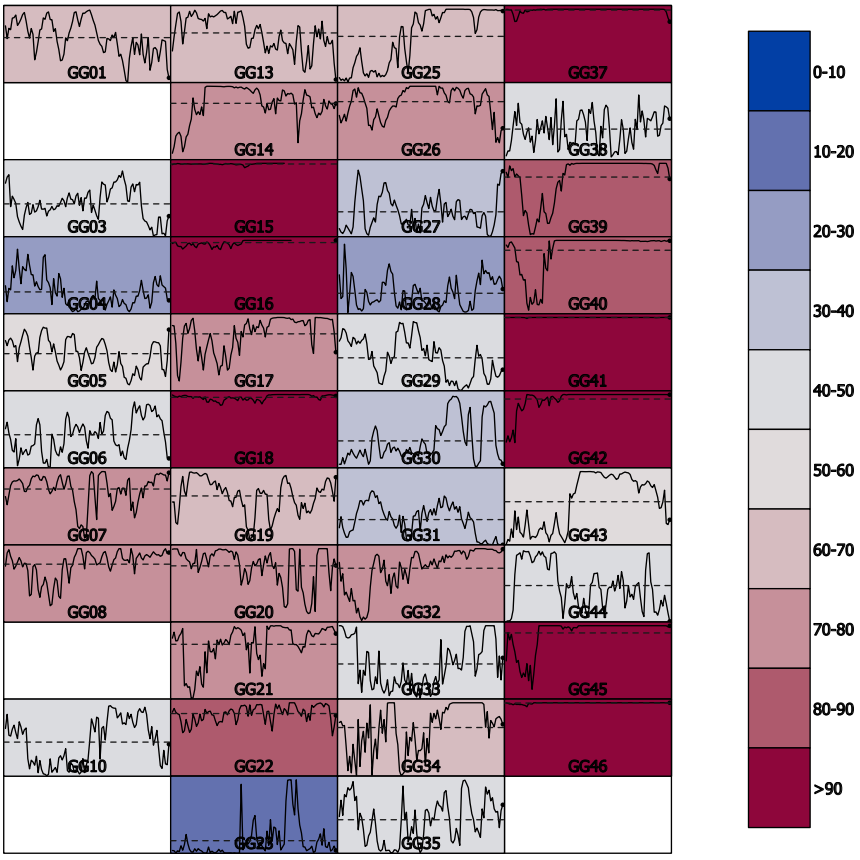


Figure 4.9: Empirical GPI graph

- The spike in the T^2-value immediately after the greatest index value occurs at time $t = 59$ as depicted in Figures 4.7 and 4.8. Principal components 2 and 4 contribute significantly to the T^2-value. From Figure 4.11f it is clear that the contribution of principal component 4 is significantly larger than the remaining components. Figure 4.11e highlights the contribution of variable S2 to the T^2-value at time $t = 59$. This result is confirmed by the PCA biplots containing principal component 4, for example Figure 4.10c.

Therefore, the combination of the variable and score contribution plots and the PCA biplots can be utilised as powerful diagnostic tools to diagnose the deviation from expected performance.

4.2.4 Summary of empirical GPI

In this section a data driven performance index (empirical GPI) was proposed. This index gives comparable results to the fundamental GPI. Figure 4.12 depicts the proposed methodology for the empirical GPI.

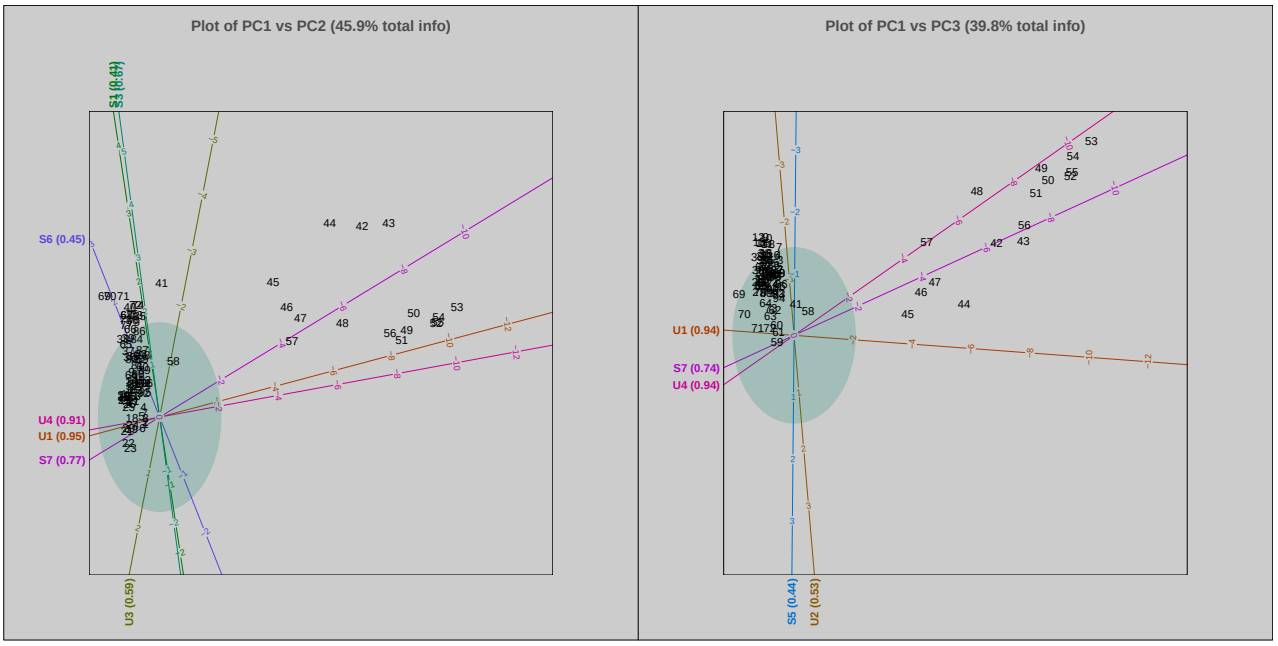
The methodology proposed here guides the user in understanding the underlying process and will highlight the interrelationships among the different variables. In addition, there are several advantages to the methodology:

- The index values are directly interpretable as the probability that the specified T^2-value is greater than or equal to the T^2-value of the reference set.
- The index yields a range between 0 and 100, where 0 is closer to the mean of the reference set, and 100 is furthest away from the mean of the reference set.
- The range of the index values is independent of the number of dimensions, and the number of variables.

There are however some disadvantages to this methodology:

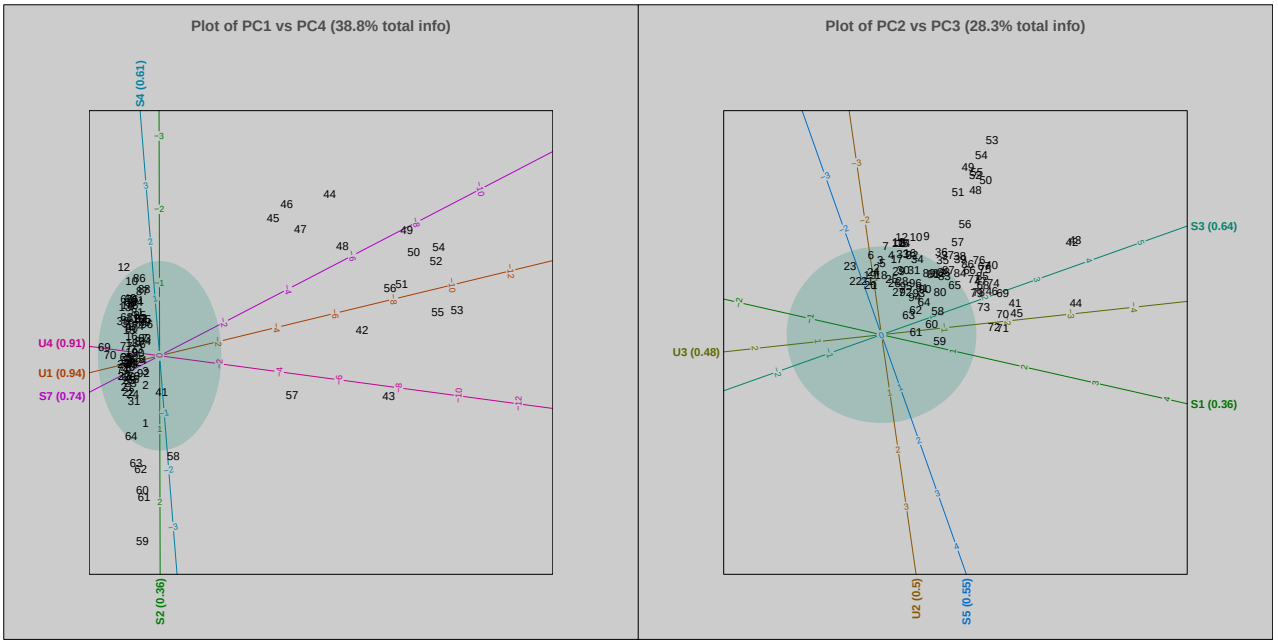
- All the variables are equally weighted. This is not necessarily a disadvantage, but the user should be aware of this property.
- Both high and low values are equally weighted for the variables. This is again not necessarily a disadvantage, but may not be optimal.
- A reference data set is required. There is however some advantage in going through the process of obtaining the reference set as valuable knowledge of the underlying process is gained.
- The empirical GPI is more computationally intensive than the fundamental GPI. However, the mathematical calculations can be stored and do not need to be calculated in real time.

These disadvantages are present in any purely empirical approach. One disadvantage specific to the gasification facility is that changes in the factory load variable are ignored. As an example, Figure 4.13 shows both the fundamental GPI and the empirical GPI for a period where the factory load was very low. The fundamental GPI (Figure 4.13a) indicates that the facility is in general performing as expected. A low factory load is generally an indication that the gasification plant is not the current bottleneck. However, the empirical GPI (Figure 4.13) incorrectly indicates that all the gasifiers are performing very poorly. Furthermore, there is no indication which of the gasifiers is truly and mostly under performing. This is a serious disadvantage that needs to be addressed before an empirical methodology can be implemented for multivariate process monitoring. In the next section a combined process and data driven approach is proposed to capture the major advantages of both of these approaches.



(a) PC1 vs PC2

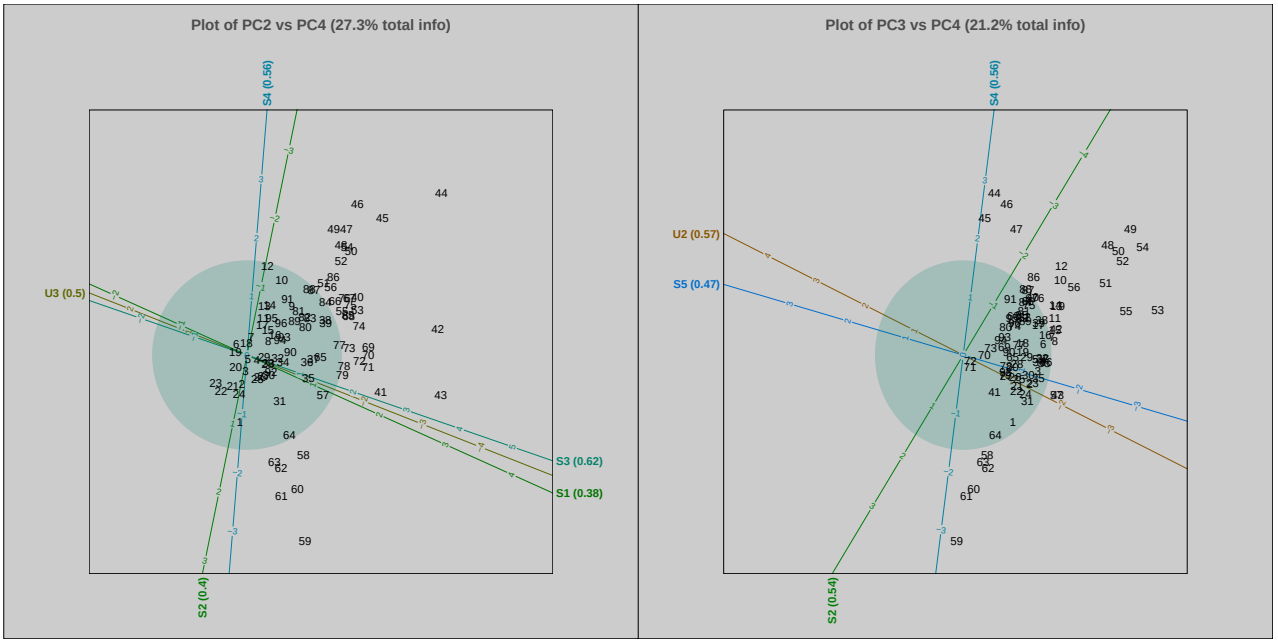
(b) PC1 vs PC3



(c) PC1 vs PC4

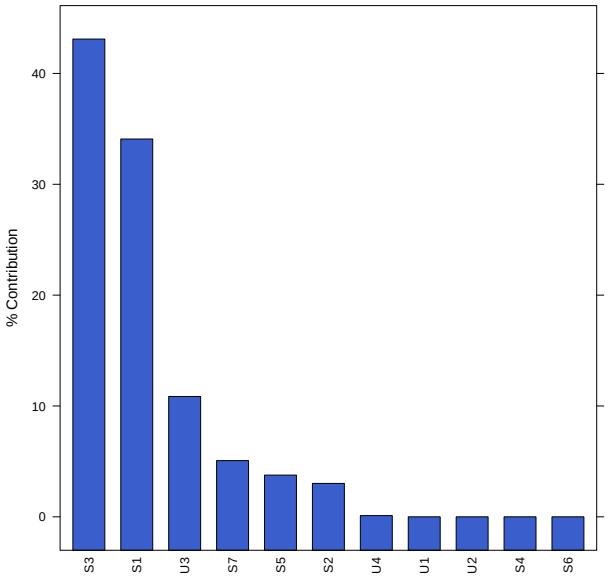
(d) PC2 vs PC3

Figure 4.10: PCA plots for different principal component combinations for GG26

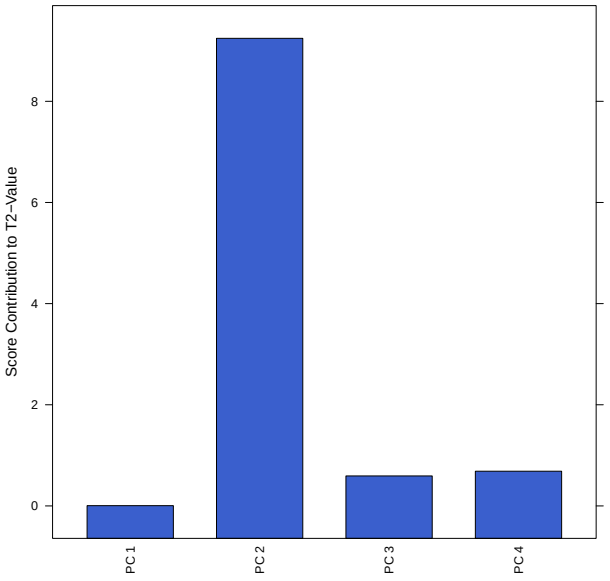


(e) PC2 vs PC4 (f) PC3 vs PC4

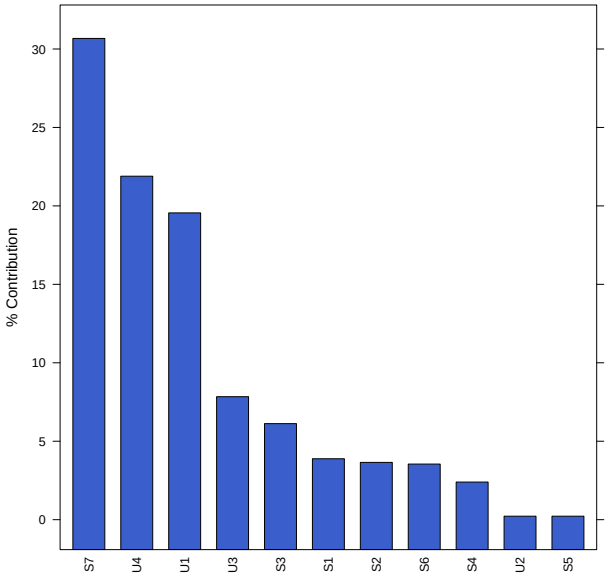
Figure 4.10: PCA plots for different principal component combinations for GG26 continued



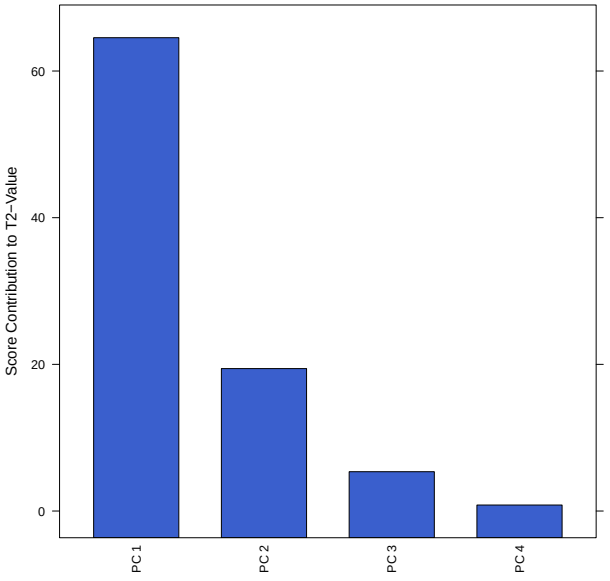
(a) Variable Contributions at $t = 41$



(b) Score Contributions at $t = 41$

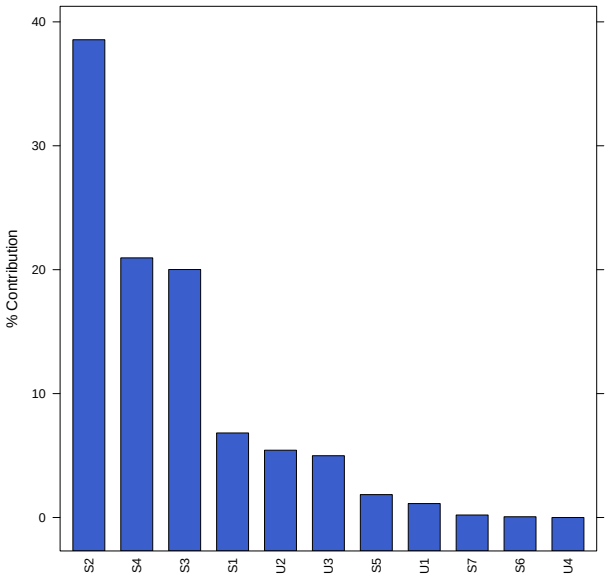


(c) Variable Contributions at $t = 43$

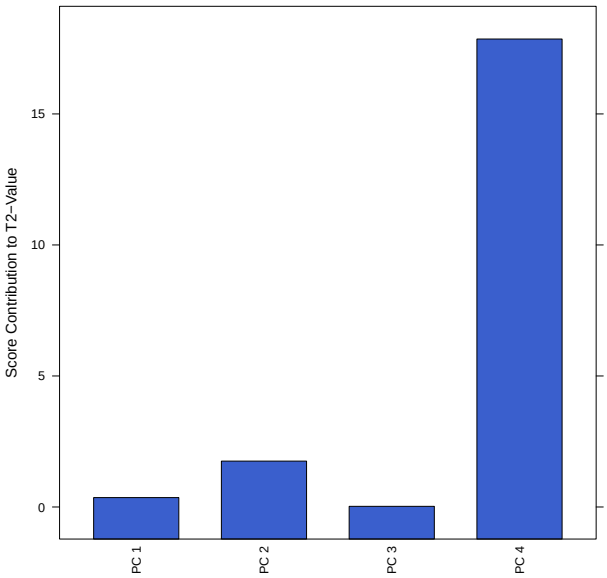


(d) Score Contributions at $t = 43$

Figure 4.11: Variable Contributions



(e) Variable Contributions at $t = 59$



(f) Score Contributions at $t = 59$

Figure 4.11: Variable Contributions continued

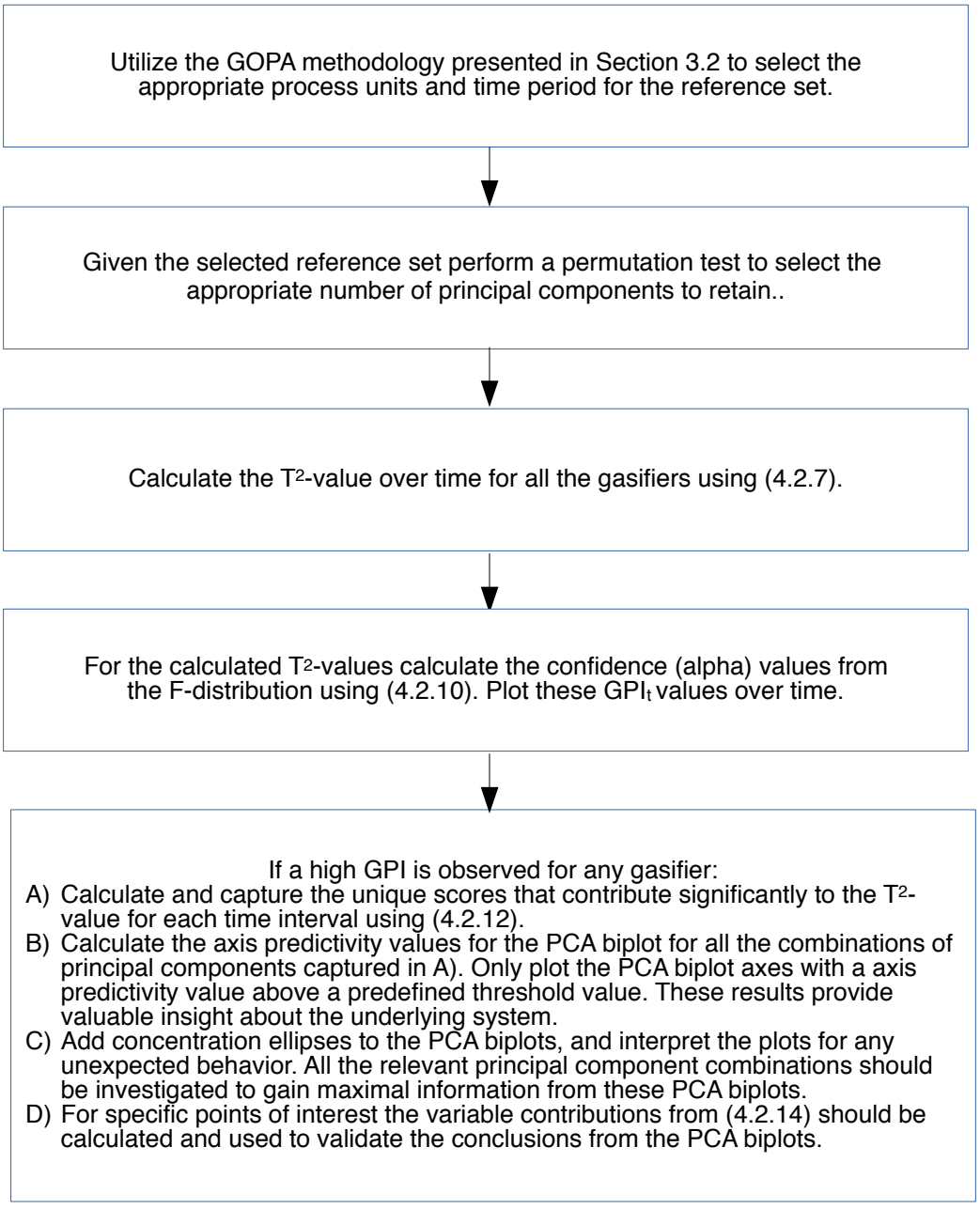
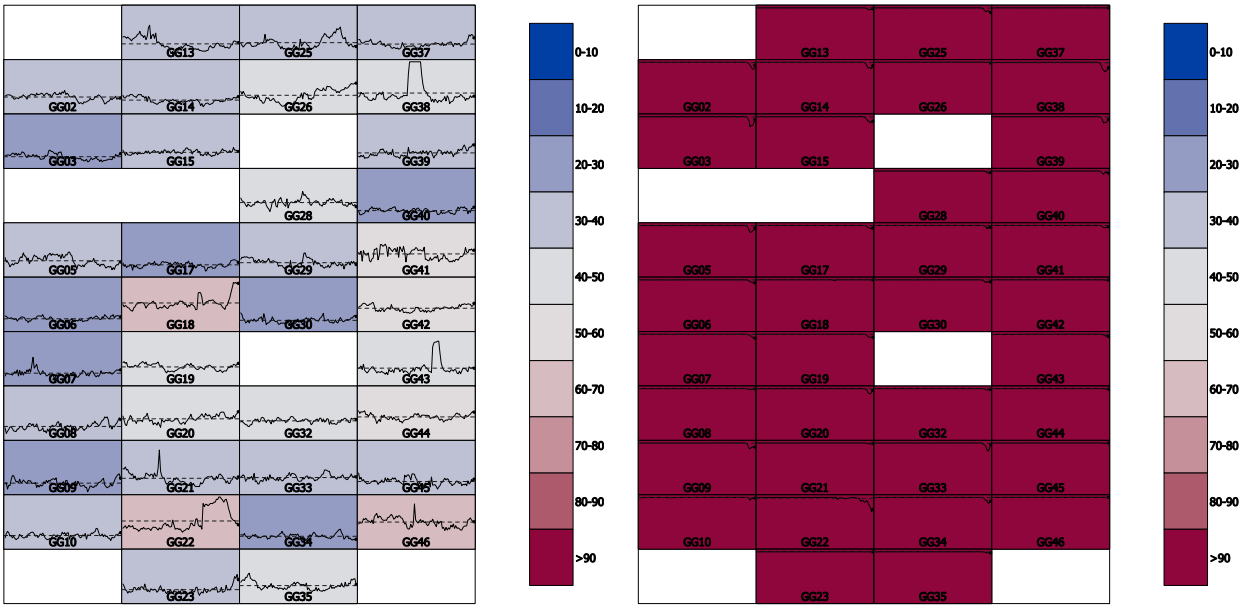


Figure 4.12: Proposed methodology for an empirical performance index



(a) Process driven GPI (b) Data driven GPI

Figure 4.13: Comparison of fundamental and empirical GPI at low factory load conditions

4.3 Integration of fundamental and empirical approaches

4.3.1 Introduction

Two different approaches to a performance index for gasification were proposed and discussed in Sections 4.1 and 4.2. Both of these approaches have certain advantages and disadvantages. It should therefore be beneficial to develop an integrated approach which combines the positive features of these two approaches. The fundamental approach is very subjective, but it does provide for some control over the weighting of variables. In addition, it allows for the dynamic adjustment of the recommended value for each variable subject to changes in a control variable (factory load in the case of the GPI). However, provision is made indirectly for different weightings of the values for being above or below the target. This is an important consideration, especially for variables such as temperature and pressure. Generally, high gas outlet temperatures can lead to instability and poor performance, but too high temperatures and pressures can lead to a safety risk. Therefore, it is important to penalise a performance index more for a higher value for these variables, than for an identical offset on the lower side.

The empirical approach is totally objective (except for the choice of variables to include) and the final weighting of the variables are obtained from the underlying mathematics of the PCA and T^2-statistics. Additionally, the process of obtaining the reference set, number of principal components to include as well as the axis predictivities lead to valuable insights into the underlying process. However, the greatest advantage of the empirical approach is that it takes into account the multidimensional character of the gasificaiton process.

4.3.2 Developing the Integrated Performance Index

Consider the variables defined in Section 4.1.2 for the fundamental GPI. To integrate the process driven and data driven performance indices, a multivariate process monitoring approach is recommended for the process deviations from the target value Opt_i. The variable z is defined as

$$z = \left(\frac{x_{ti} - Opt_{ti}}{\text{ifelse}[[x_{ti} - Opt_{ti}] < 0, Dmin_{ti}, Dmax_{ti}]} \right) \tag{4.3.1}$$

Two adjustments were made to the numerator of (4.1.9) to obtain the definition of z:

- The absolute value was removed to retain the direction of deviation from the recommended value.
- The recommended value is subtracted from the current value to map a positive z_{ti} value to $x_{ti} \geq Opt_{ti}$ to aid in the interpretation of the results.

Let $\boldsymbol{Z}$ be the $n \times p$ matrix of z_{ti} values. The t-th row of $\boldsymbol{Z}$ is $\boldsymbol{z}_t = z_{t1}, \cdots, z_{tp}$ for $t = 1, \cdots, n$. Therefore, $\boldsymbol{Z}$ can be substituted for $\boldsymbol{X}$ in the multivariate monitoring methodology discussed in Chapter 3. In addition, the empirical index developed in Section 4.2.2 can be directly applied to $\boldsymbol{Z}$. Therefore, in substituting $\boldsymbol{Z}$ for $\boldsymbol{X}$ in the multivariate monitoring methodology, the process deviation from the target value Opt_i, which is dynamically adjusted for the control variable m, is monitored. In the next section the selection of a reference set for the integrated approach, utilising the methodology developed in Section 3.2, will be discussed and demonstrated. In Section 4.3.3.3 the number of principal components to retain for the reference data set, as well as the axis predictivity will be discussed. In Section 4.3.4 the reference set together with the scaling in this section will be used to demonstrate the integrated GPI. The use of PCA biplots will be demonstrated to interpret causes for deviations from expected performance. Lastly, the applications of CVA biplots to $\boldsymbol{Z}$ will be discussed and demonstrated in Section 4.3.5 .

4.3.3 Reference Set Selection

4.3.3.1 Introduction to reference set selection for the integrated approach

In Section 3.2 a methodology was proposed for applying GOPA for the selection of a reference set for the multivariate process monitoring of multiple production processes. Now, from (3.2.8) and (3.2.9)

$$\boldsymbol{G} = \frac{1}{K} \sum_{k=1}^{K} s_k \boldsymbol{Z}_k \boldsymbol{B}_k \tag{4.3.2}$$

and the norm

$$\sum_{k=1}^{K} \| s_k \boldsymbol{Z}_k \boldsymbol{B}_k - \boldsymbol{G} \|^2 \tag{4.3.3}$$

is minimised in the GOPA analysis. The reference set selection process will be demonstrated in the next section.

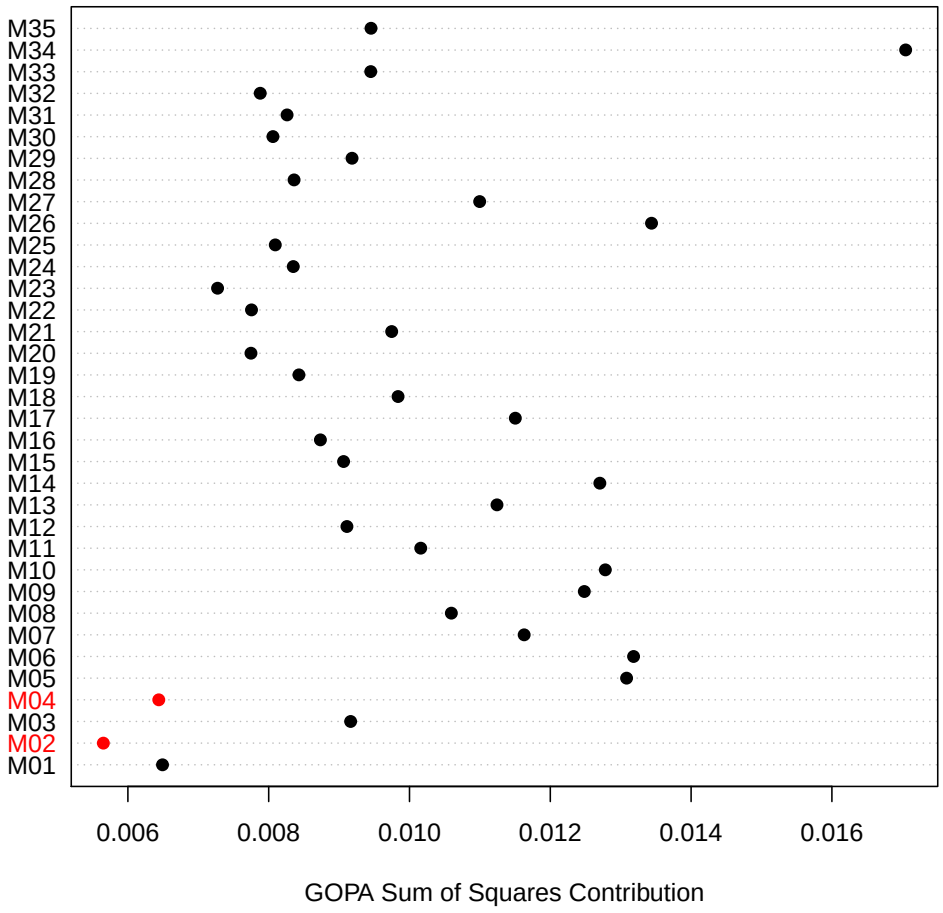


Figure 4.14: Average monthly GOPA sum of squares contributions for the Western factory

4.3.3.2 Reference set selection for the integrated performance index

Similar to the discussion in Section 3.2.4 the GOPA optimization was performed for the Western and Eastern factories respectively. The Average monthly GOPA sum of squares contributions are depicted in Figures 4.14 and 4.15. The optimal first two months are highlighted in red. The optimal two months for the Western factory are again M02 and M04 and those for the Eastern factory M20 and M24 as tabulated in Table 4.2. The similarity of the results for the scaled offset from the recommended values to the unscaled results could be an indication that the underlying structure of the data has not changed by using $\boldsymbol{Z}$ as the GOPA algorithm removes the superfluous differences (see Section 3.2.3).

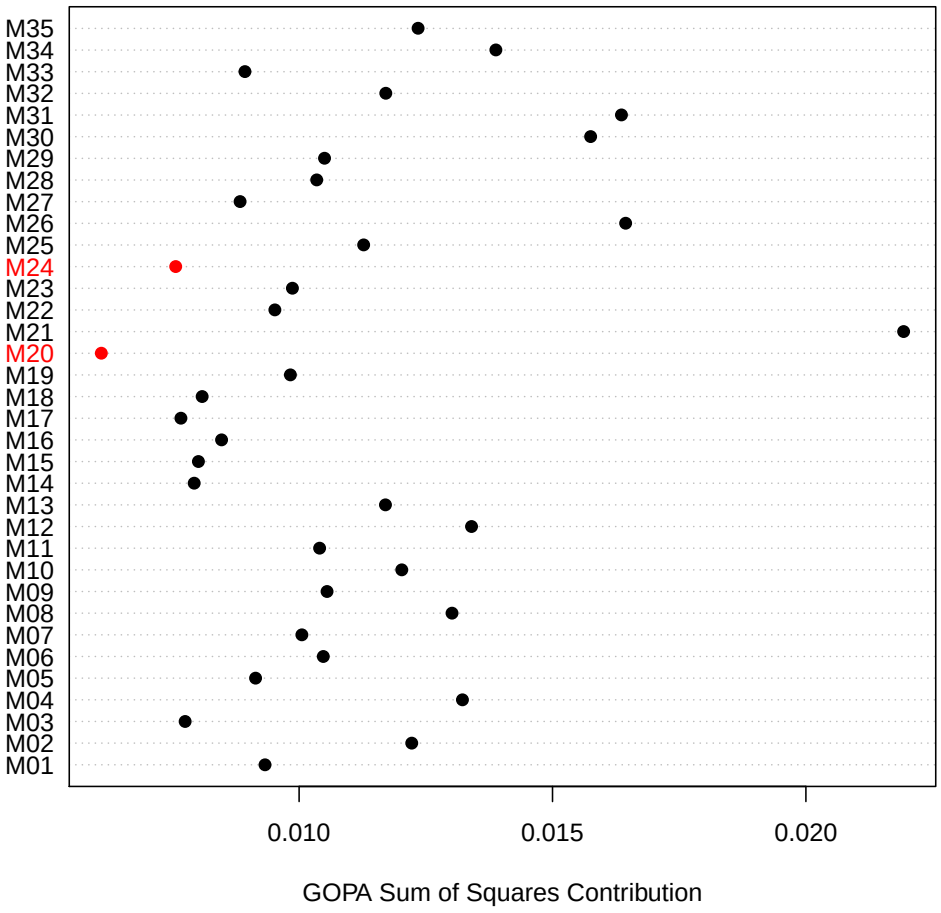


Figure 4.15: Average monthly GOPA sum of squares contributions for the Eastern factory

Table 4.2: Optimal combination of Months

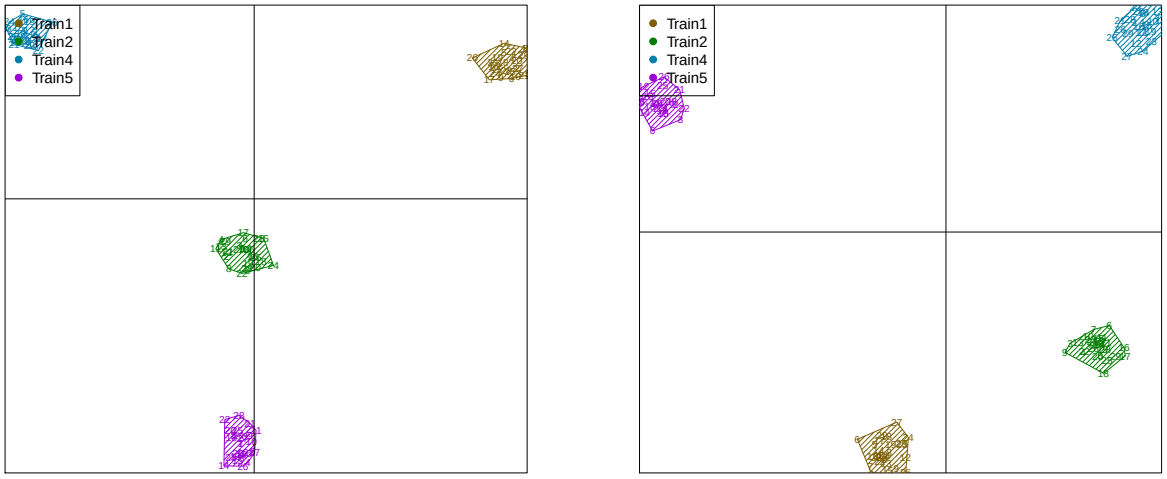
Side	Month 1	Month 2
West	M02	M04
East	M20	M24

Given the optimal reference set, the selection of the optimal train from the output of the GOPA analysis can proceed. The group average configuration $\boldsymbol{G}$ obtained from the GOPA output can be used to select the train closest to the overall centroid $\boldsymbol{O}$ (see Section 3.2.4.1). Figures 4.16a and 4.16b depict the PCA plots of the group averages including the variation for each train for the optimal one month period for the Western and Eastern factory respectively. The actual calculated Euclidean distances of the trains from the overall cen-

troid $\boldsymbol{O}$ are provided in Table 4.3. For the Western factory it is observed from Figure 4.16a that Train Two is closest to the overall centroid. This result is confirmed by the Euclidean distances in Table 4.3. For the Eastern factory however, Figure 4.16b would lead to the selection of Train Two as the optimal train, whereas from Table 4.3 it is clear that Train One is closest to the overall centroid. The Euclidean distances for Train One and Train Two are however very similar, and although the quality of the two dimensional display is high (69.63% as depicted in Table 4.4), it is still only an approximation of the full dimensional space. As is true for all multivariate projection methods in general the visual display is a powerful guide to the underlying structure but it is always advised to confirm the final results with the underlying algebra.

Table 4.3: Euclidean distance of trains from overall centroid $\boldsymbol{O}$.

Side	Months	TR1	TR2	TR4	TR5
West	1	0.5122	0.4802	0.5091	0.4930
East	1	0.4829	0.4899	0.5009	0.5211



(a) Optimal one month for Western factory (b) Optimal one months for Eastern factory

Figure 4.16: PCA plots of group averages including the variation for each train for the optimal one and two month combinations

In conclusion, the GOPA methodology developed and proposed in Section 3.2 was applied to the scaled data set $\boldsymbol{Z}$. The optimal reference periods and trains were identical to the results in Section 3.2.4.1. For the Western factory,

Table 4.4: Biplot quality of the two-dimensional PCA plot for different sides (Figures 4.16a - 4.16b)

Side	One Month
West	69.20%
East	69.63%

Train Two for the period of February 2012 will be utilized as reference data set, and for the Eastern factory, Train One for the period M20 will be utilized as reference data set.

In the next section the optimal number of principal components to retain as well as the axis predictivity will be calculated for the reference set for the Eastern factory. Although only the Eastern factory is discussed the results for the full facility will be demonstrated in Chapter 6.

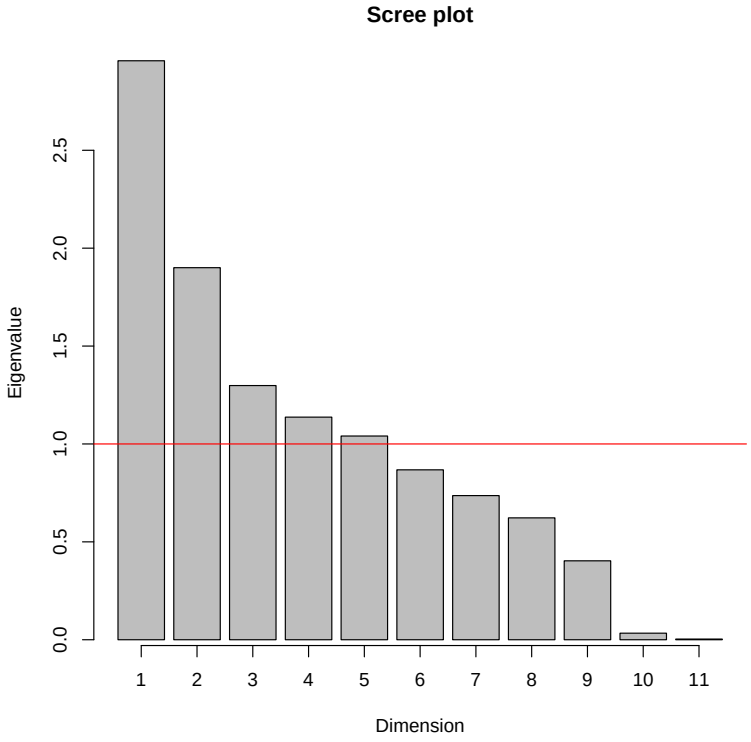


Figure 4.17: Scree Plot for reference data ($\boldsymbol{X}_*$) for Eastern factory

4.3.3.3 Number of principal components to include and axis predictivity results

In Section 3.3.4.1 various strategies are discussed for determining the principal components to retain. The eigenvalues of the correlation matrix for the reference set are calculated. The results are shown in Figure 4.17. According to this criterion, five principal components should be retained.

Greenacre and Primicerio (2014) page 223-224 discuss a formal methodology to find the number of significant principal components using a permutation test. The permutation test was performed for $j = 9999$ for the eigenvalues. All the values of the permuted data sets were smaller than those for the original data set up to and including five dimensions. For six and higher dimensions all the values of the permuted data sets were higher than the original data set. Therefore, five principal components should be retained.

The axis predictivities are provided in Table 4.6 for all the principal component combinations up to and including five dimensions. These values correspond to the results in Section 3.3.4.2 Table 3.7 except for the inclusion of the fifth principal component. The PCA loadings for the first five principal components are provided in Table 4.7. Closer investigation of Tables 4.6 and 4.7 leads to the following observations:

Table 4.5: PCA Biplot Quality

	PC (x-axis)	PC (y-axis)	Quality (%)
Figure 4.18a	1	2	44.16
Figure 4.18b	1	3	38.69
Figure 4.18c	1	4	37.22
Figure 4.18d	1	5	36.34
Figure 4.18e	2	3	29.08
Figure 4.18f	2	4	27.61
Figure 4.18g	2	5	26.73
Figure 4.18h	3	4	22.14
Figure 4.18i	3	5	21.26
Figure 4.18j	4	5	19.80

Table 4.6: Axis Predictivity

	PC1	PC2	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1	1	2	0.94	0.13	0.57	0.42	0.03	0.68	0.02	0.11	0.18	0.88	0.90
2	1	3	0.95	0.62	0.07	0.05	0.07	0.03	0.05	0.30	0.21	0.95	0.96
3	1	4	0.94	0.19	0.09	0.01	0.45	0.01	0.56	0.04	0.05	0.88	0.89
4	1	5	0.94	0.16	0.08	0.20	0.21	0.00	0.16	0.14	0.35	0.87	0.88
5	2	3	0.02	0.49	0.50	0.46	0.10	0.70	0.06	0.35	0.35	0.09	0.09
6	2	4	0.01	0.07	0.51	0.42	0.48	0.68	0.57	0.08	0.19	0.01	0.02
7	2	5	0.01	0.04	0.50	0.61	0.24	0.68	0.17	0.19	0.49	0.01	0.01
8	3	4	0.02	0.55	0.01	0.05	0.52	0.03	0.60	0.27	0.22	0.08	0.08
9	3	5	0.02	0.52	0.00	0.24	0.27	0.03	0.20	0.38	0.52	0.08	0.08
10	4	5	0.01	0.10	0.02	0.20	0.65	0.01	0.71	0.11	0.36	0.00	0.01

Table 4.7: PCA loadings for first five principal components

	PC1	PC2	PC3	PC4	PC5
U1	-0.56	-0.05	0.09	0.06	0.08
U2	0.21	-0.04	-0.61	-0.24	-0.18
U3	-0.16	0.51	0.01	-0.11	-0.06
S1	0.03	-0.47	-0.19	0.06	0.44
S2	-0.03	0.13	-0.23	0.63	-0.44
S3	0.01	-0.60	0.14	0.07	-0.04
S4	-0.04	-0.08	0.18	-0.70	-0.39
S5	0.11	-0.20	-0.46	-0.06	-0.32
S6	-0.08	-0.29	0.38	0.16	-0.56
S7	-0.54	-0.07	-0.24	-0.06	-0.02
U4	-0.55	-0.08	-0.25	-0.07	-0.01

1. The axis predictivities for variables U1, S7 and U4 are higher in all combinations where principal component one is present. This observation is confirmed by the loadings for principal component one, where all three these variables have relatively large loading values compared to the other variables. U1, S7 and U4 are all variables directly linked to reactor load. All the loadings are negative, and therefore in the same direction. It is important to remember that the values of the loadings are unique up to multiplication by -1. Therefore the relative (but not absolute) directions of the variables can be interpreted.

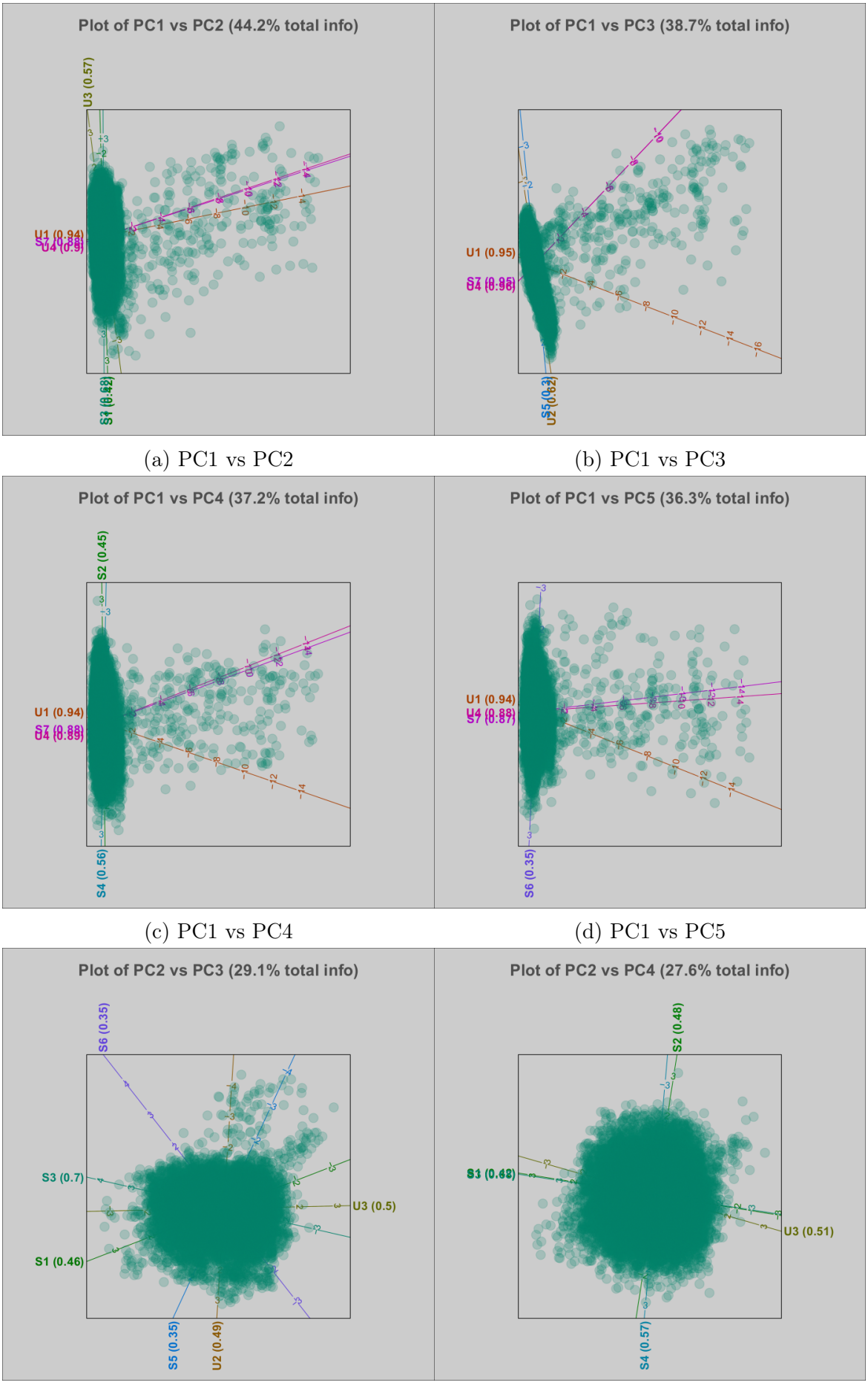
2. The axis predictivities for U3, S1 and S3 are higher in all combinations where principal component two is present. This observation is confirmed by the loadings for principal component two. Variable U3 has a relatively large positive loading, and variables S1 and S3 have relatively large negative loadings.

3. The axis predictivities for U2 and S5 are higher in all combinations where principal component three is present. This observation is confirmed by the loadings for principal component three. Both variable U2 and S5 have relatively large negative loadings.

4. The axis predictivities for S2 and S4 are higher in all combinations where principal component four is present. This observation is confirmed by the loadings for principal component four.

5. The axis predictivity for S6 is higher in all combinations where principal component five is present. Interestingly principal component five exhibit relatively high loadings for S1, S2, S4, S5 and S6. It could therefore be concluded that the values for S6 are not independent of these other variables.

It can therefore be concluded that the structure in the principal components successfully captured the underlying structure of the gasification process. Additionally the reference set selected captures the process conditions, and should therefore be appropriate for monitoring the process. The scaling of the data retained the underlying structure of the process. A cutoff value of 0.3 will be utilized to include all the relevant axes for each principal component combination in the study (as highlighted in gray in Table 4.6). Figure 4.18 depicts the PCA biplots for $\boldsymbol{Z}$ for all the relevant principal component combinations. These PCA biplots will be used as scaffolding for the monitoring biplots discussed in the next section.



(a) PC1 vs PC2

(b) PC1 vs PC3

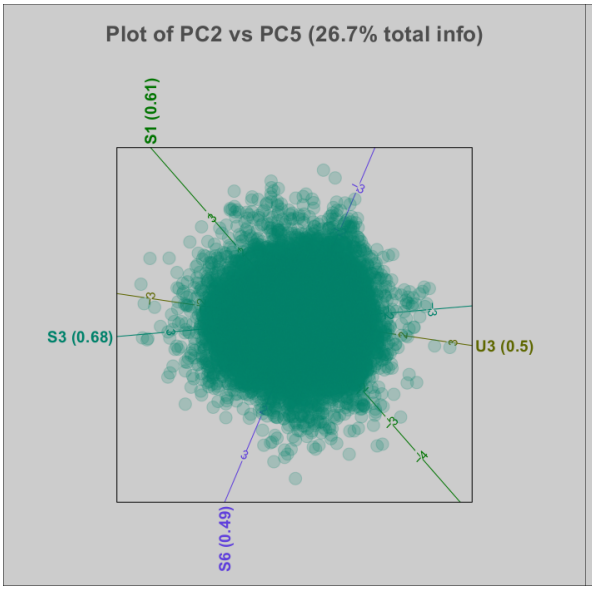
(c) PC1 vs PC4

(d) PC1 vs PC5

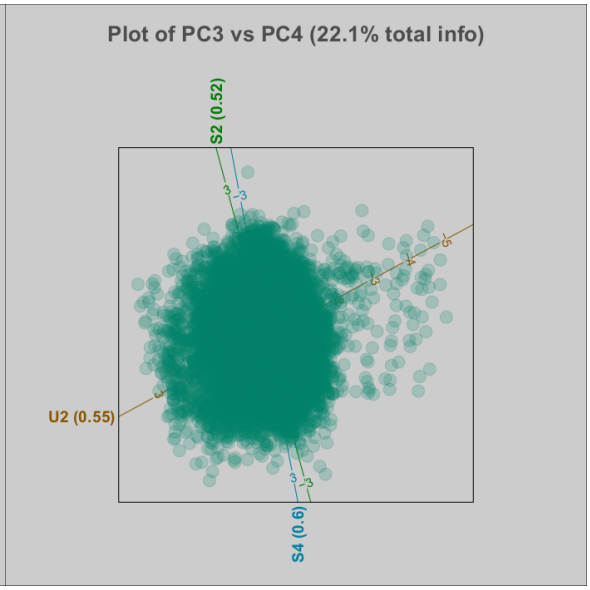
(e) PC2 vs PC3

(f) PC2 vs PC4

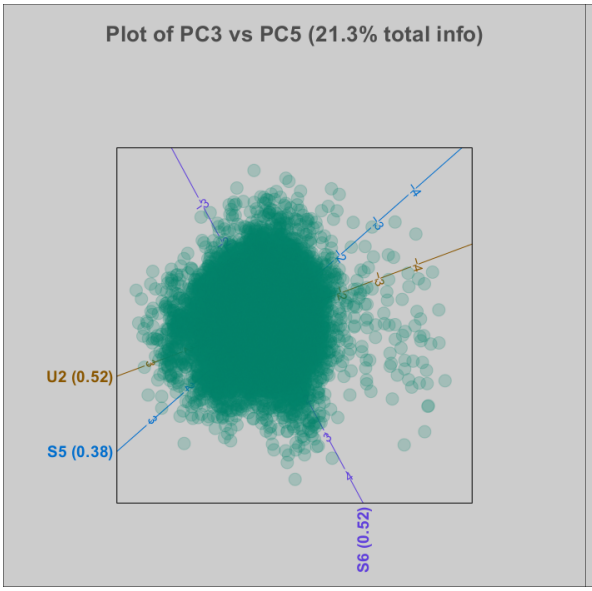
Figure 4.18: PCA plots of different principal component combinations for the reference set, with axes chosen according to the axis predictivity criterion



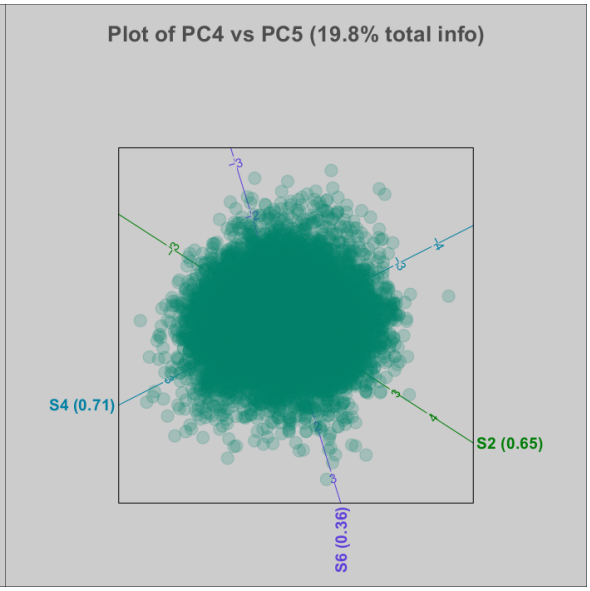
(g) PC2 vs PC5



(h) PC3 vs PC4



(i) PC3 vs PC5



(j) PC4 vs PC5

Figure 4.18: PCA plots of different principal component combinations for the reference set, with axes chosen according to the axis predictivity criterion continued

4.3.4 Integrated Gasifier Performance Index (GPI)

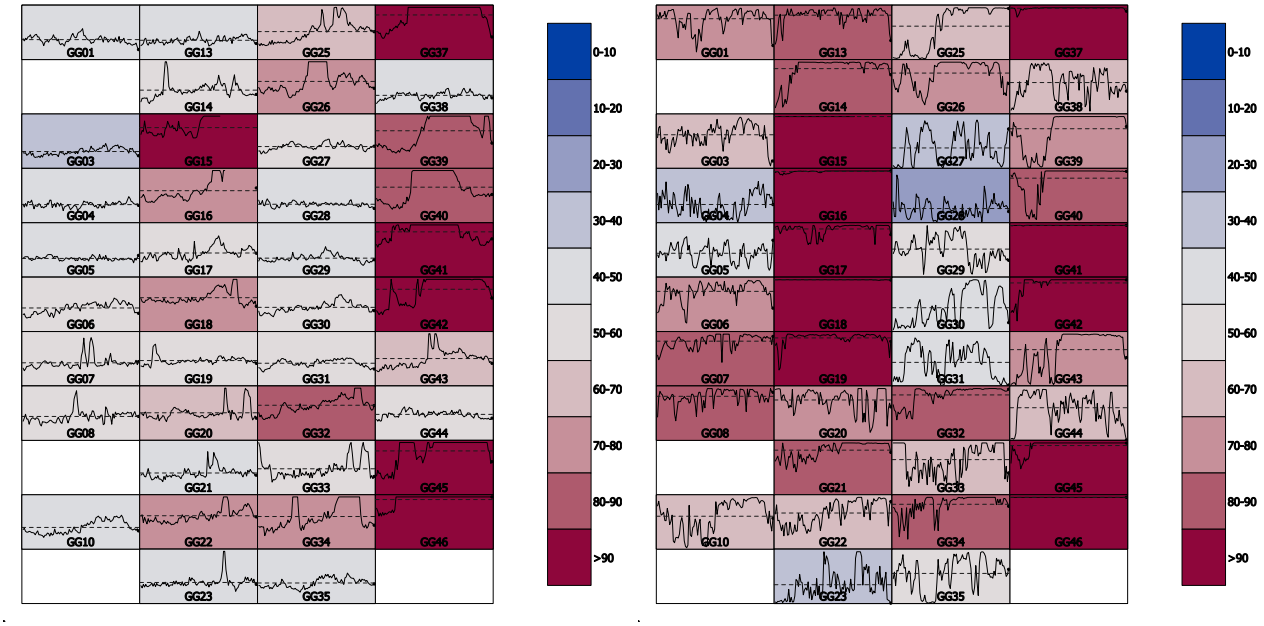
Consider the variables defined in Section 4.2.2 for the fundamental GPI. The empirical GPI is performed as discussed in Section 4.2.3. From (4.2.7), let

$$\frac{1}{\sqrt{n-1}}\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{4.3.4}$$

If $\boldsymbol{z}$ is defined as the scaled offset from the recommended target values for a new observation and setting $r^* = 5$ as the optimal number of principal components to retain, the T^2-statistic for $\boldsymbol{z}$ is

$$T^2 = \boldsymbol{z}^T \boldsymbol{V}_{r^*} (\boldsymbol{\Sigma}_{r^*}^T \boldsymbol{\Sigma}_{r^*})^{-1} \boldsymbol{V}_{r^*}^T \boldsymbol{z} \tag{4.3.5}$$

From equations (4.2.8) - (4.2.10) we calculate the α confidence values for $\boldsymbol{z}$ (4.2.10) for the T^2-values. These values are now used as a performance index. The integrated GPI values are calculated on 15 minute time weighted average data for all the variables in Table 4.1 for a 24 hour period, and displayed on the same custom graph (heatmap) as the fundamental and empirical GPI. Refer to Figure 4.19, which shows a snapshot for both the fundamental GPI and the integrated GPI for the Eastern factory. Both methods flags similar gasifiers for servere performance deviation, except for some very high index values on Train 2 for the integrated GPI. These gasifiers will now be investigated for the contributing variables to the high integrated GPI values.



(a) Process driven GPI (b) Integrated GPI

Figure 4.19: Comparison of fundamental and integrated GPI

Figure 4.20 depicts monitoring PCA biplots for all the principal component combinations up to five components for GG26 for the specific period under investigation. As discussed in Section 4.2.3 the score contributions can be utilised to filter the principal components for the biplots. However, all the principal component scores were significant for at least one of the time intervals (t), and therefore all five were included in this analysis. GG26 was again chosen to compare the monitoring biplots performed on the scaled reference set $\boldsymbol{Z}$ with the biplot in Figure 4.10 utilising the purely data driven approach. All the biplots including PC1 (Figures 4.20a to 4.20d) again highlight the period of low gasifier load as discussed in Section 4.1.3. Additionally all the biplots containing PC4 again indicates the high degree of variability in variables S2 and S4. Figure 4.20e highlights the high values for variable S3.

However, Figure 4.20e also yields an interesting contradiction. Very low values for variables S5 and U2 are indicated during the exact same time period when variable U2 had very high values (see Figure 4.21). These contradicting results are present in all the biplots containing PC3. Referring to Figure 4.24, which includes all the axes, it can be observed that the low values for variables S5 and U2 are in the direction of very low U4 and S7 values. Referring to Figures 4.22 and 4.23, the values for these two variables are far below even the minimum value, and it can therefore be concluded that even though the predictivity value for these two values are very low for Figure 4.20e, the effect of the large deviation during this period still projects in the direction of decreasing values onto these two axes. This result is confirmed by the variable contribution plot in Figure 4.25 at time $t = 53$. The contributions of U4 and S7 account for almost 80% of the total T^2-value at time $t = 53$. These results again highlight the importance of the careful analyses of the results from the multivariate projection methods, as it can lead to significant insights. Additionally, it is clear that the PCA biplots constructed from the scaled offset from the recommended values give comparable results to the purely data driven results for normal factory load values.

Referring to Figure 4.19, the biggest discrepancies between the two different approaches are highlighted for GG17. The PCA biplots for the different principal components for GG17 are depicted in Figure 4.26, and the trend plots for the different variables are depicted in Figure 4.27. A quick overview of Figure 4.26 immediately highlights high values for variables U2 and S5. From Figures 4.27b and S5 4.27h it can be observed that variables U2 and S5 are indeed higher than the recommended values. Variable S5 is a measured value that is prone to inaccuracies. Variable U2 however is a control variable that will generally have lower variability and should therefore not undergo frequent changes.

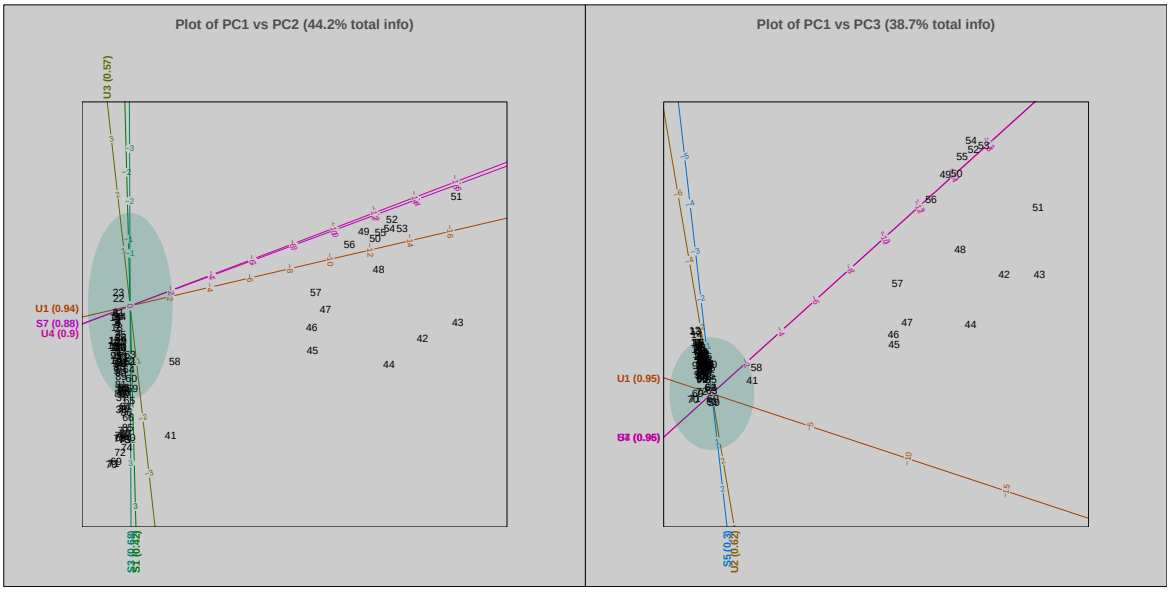
From Figure 4.27b it is clear that the variable U2 was adjusted to a higher value at least two times during this 24 hour period. As variable U2 will contain low variability in the reference data set $\boldsymbol{Z}$, these changes will have a very low probability of occurring and therefore will result in a high T^2 and correspond-

ing confidence value. For 70 out of the 96 values variable U2 was either the highest or second highest contributing variable to the T^2-value. Further investigation of the other gasifiers on Train 2 revealed that variable U2 was set to high values for all of the gasifiers except GG22 and GG23. These high values will explain the discrepancy between the process driven GPI and integrated GPI. It can be speculated that these high values should in fact be highlighted as it is important process information which the engineers should be aware of. The integrated GPI is therefore comparable to the fundamental GPI, and in the cases where discrepancies occurred the integrated GPI highlighted important information that would not have been apparent if only the fundamental approach was utilised.

One of the shortcomings of the empirical approach previously highlighted was its failure to adapt to changes in the factory load variable. One of the main goals of the integrated GPI was to be able to dynamically adapt to these changes. The fundamental and empirical GPI approaches were compared in Figure 4.13 for a period of low factory load values, and it was concluded that the empirical approach was not useful in periods of low factory load conditions. In the first part of this section it was demonstrated that during periods of normal factory load the integrated GPI performed as well as the empirical and the fundamental approaches. It will now be demonstrated that the integrated approach performs as well as the fundamental approach in periods of low factory load.

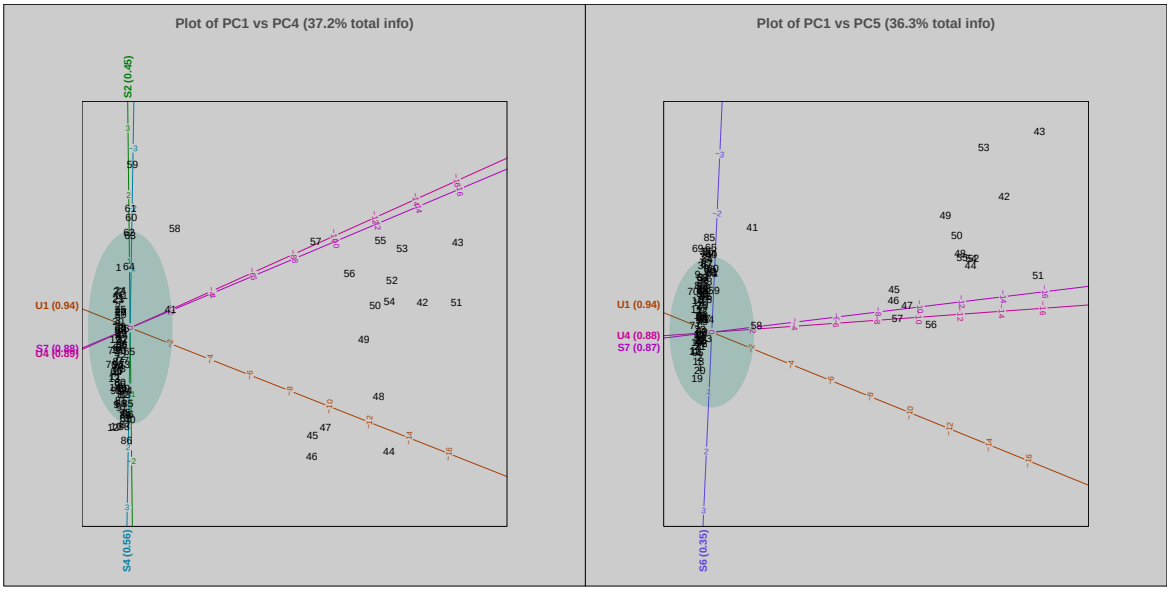
In Figure 4.28 both the integrated GPI and the fundamental GPI are depicted for the same period as in Figure 4.13. It is clear that in contrast to the empirical GPI, the integrated GPI successfully adapts to the low factory load. Additionally, it is still possible to observe gasifiers which deviate from expected performance. For example, from Figure 4.28 GG46 has a very high integrated GPI value. The PCA biplots for GG46 are provided in Figure 4.29 and the trend plots in Figure 4.30. From Figure 4.29b it is clear that variables U2 and S5 are higher than expected. This can be confirmed in all the plots containing PC3. It can be confirmed from Figure 4.30b that the values for U2 are higher than the recommended value. However, from Figure 4.30h it can be observed that the S5 values are extremely high, and would therefore also have a big impact on the index value. The high S5 value could be due to measurement error which should be investigated. These results are confirmed by Figure 4.31 which depicts the frequency of the variables occurring as one of the two highest contributing variables. These results demonstrate that the integrated GPI successfully adapts to the lower factory load values, and are capable of detecting deviations from expected performance.

In the next section the application of CVA biplots to the scaled data will be discussed and demonstrated.



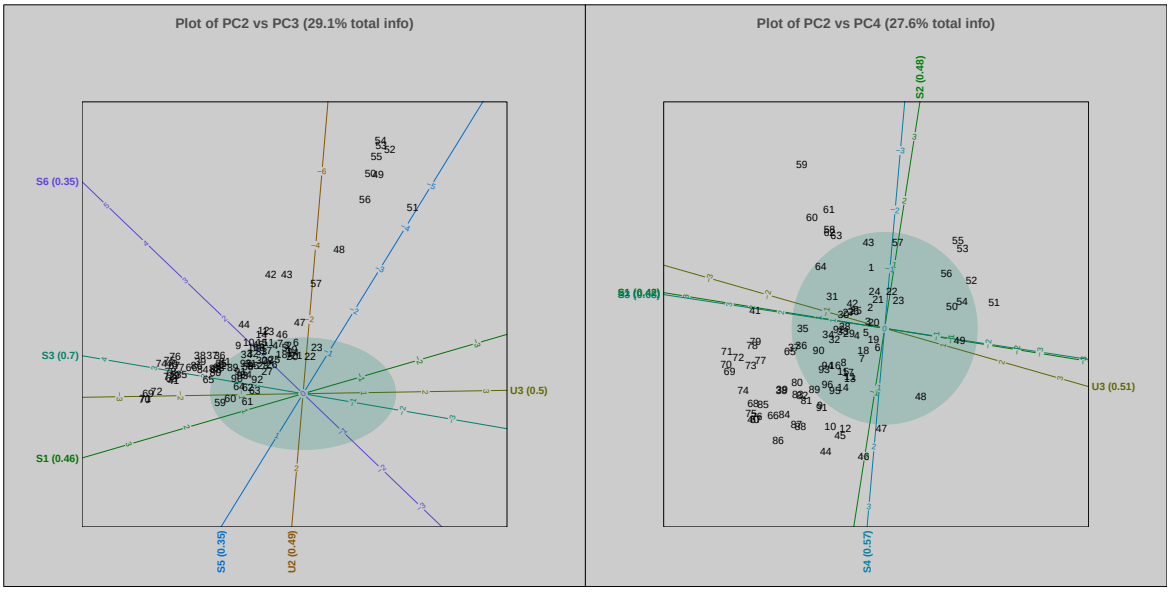
(a) PC1 vs PC2

(b) PC1 vs PC3



(c) PC1 vs PC4

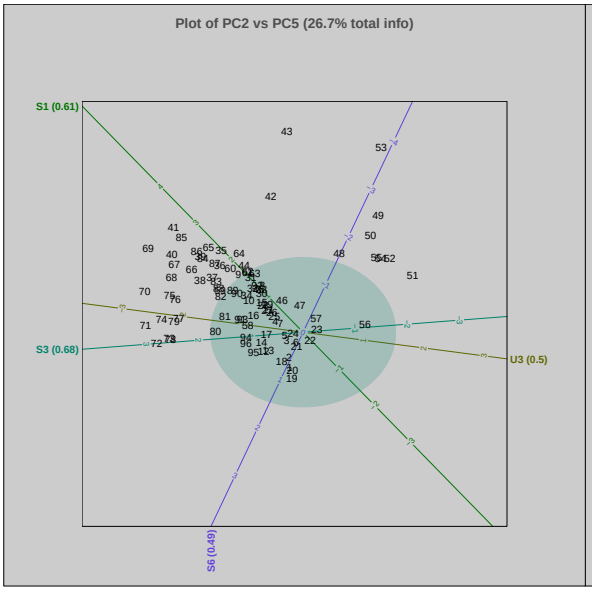
(d) PC1 vs PC5



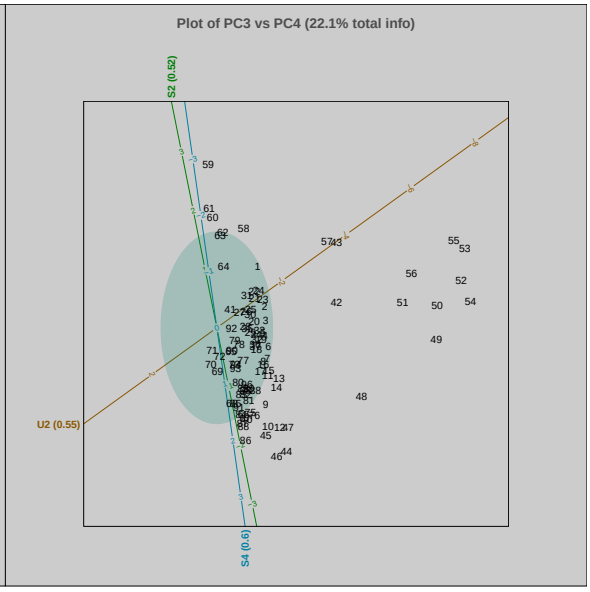
(e) PC2 vs PC3

(f) PC2 vs PC4

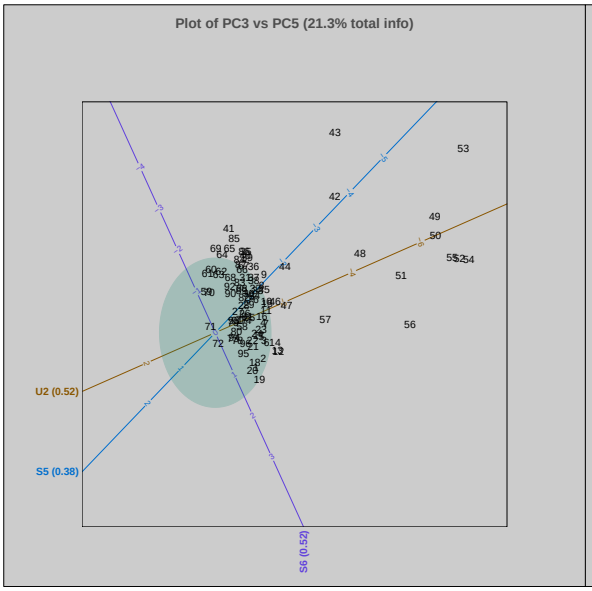
Figure 4.20: PCA plots of different principal component combinations for GG26



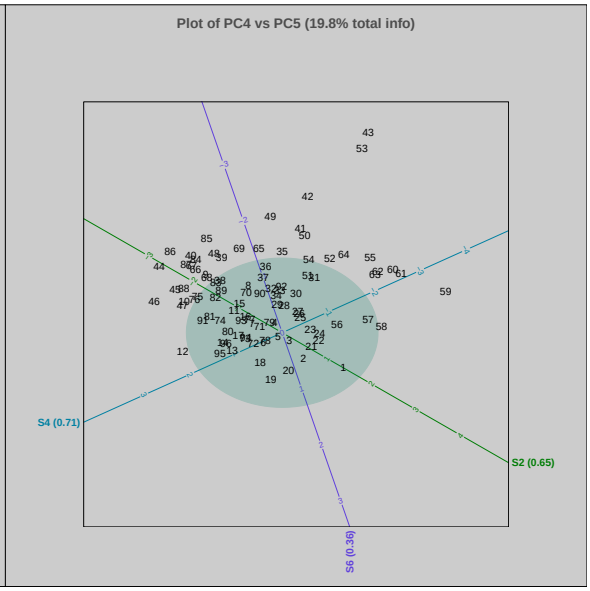
(g) PC2 vs PC5



(h) PC3 vs PC4



(i) PC3 vs PC5



(j) PC4 vs PC5

Figure 4.20: PCA plots of different principal component combinations for GG26 continued

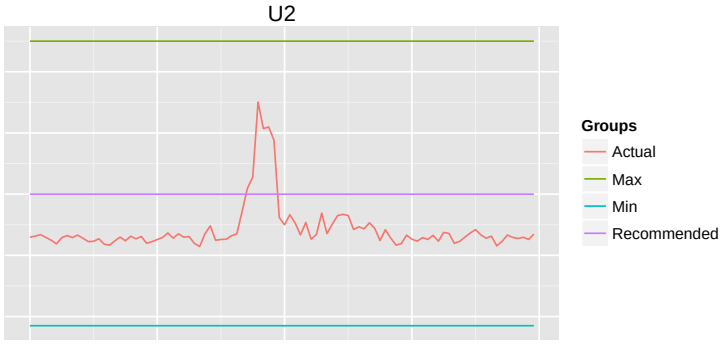


Figure 4.21: Trend plot for GG26 variable U2

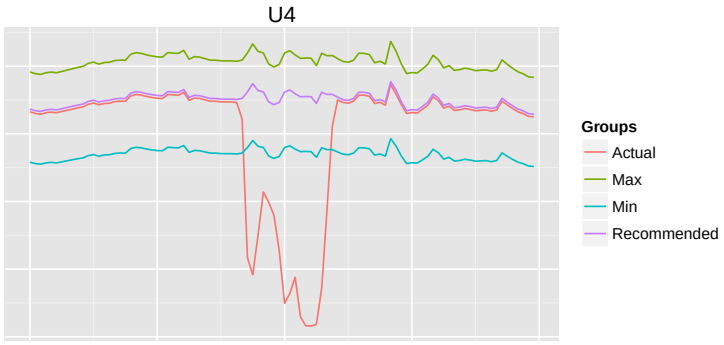


Figure 4.22: Trend plot for GG26 variable U4

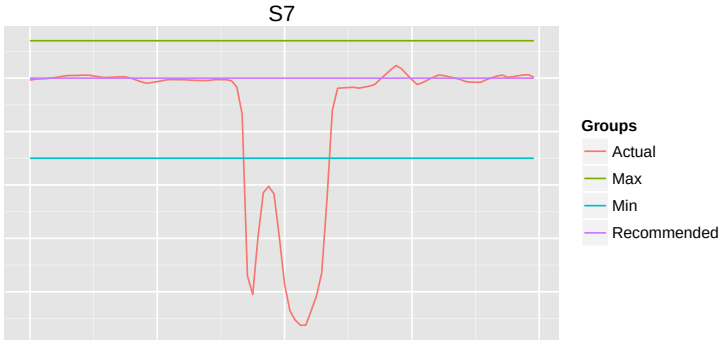


Figure 4.23: Trend plot for GG26 variable S7

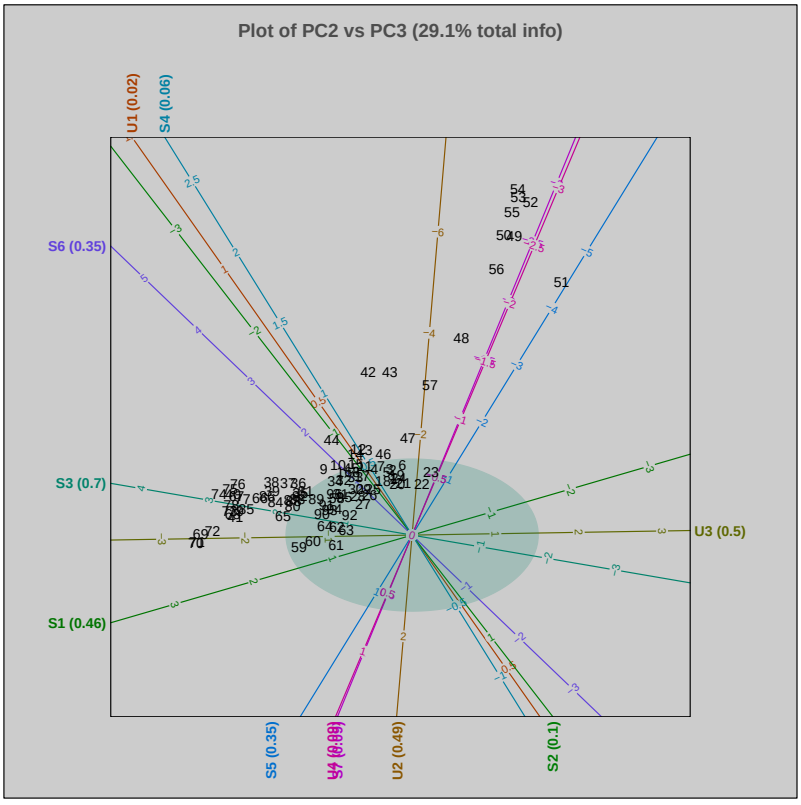


Figure 4.24: Monitoring biplot for with PC2 vs PC3 including all axes for GG26

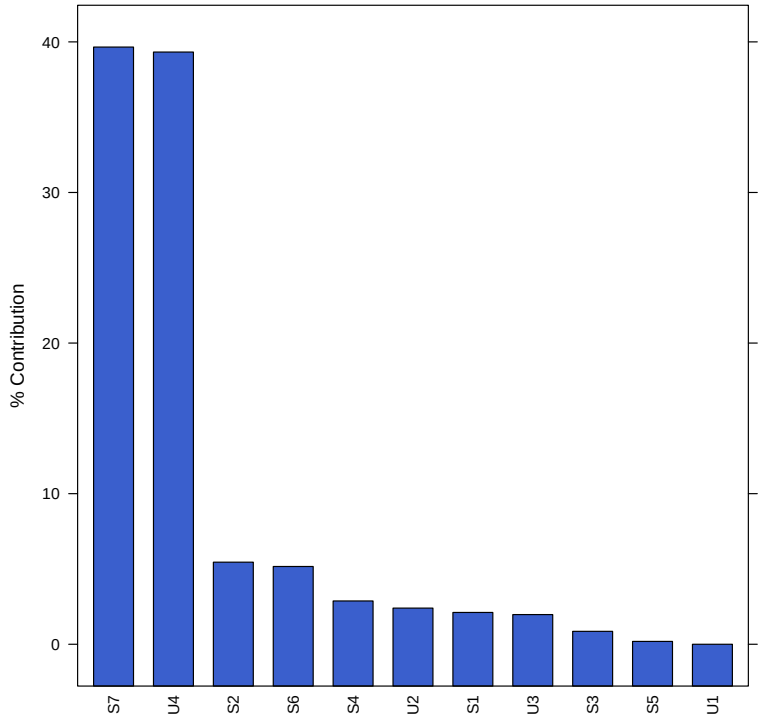
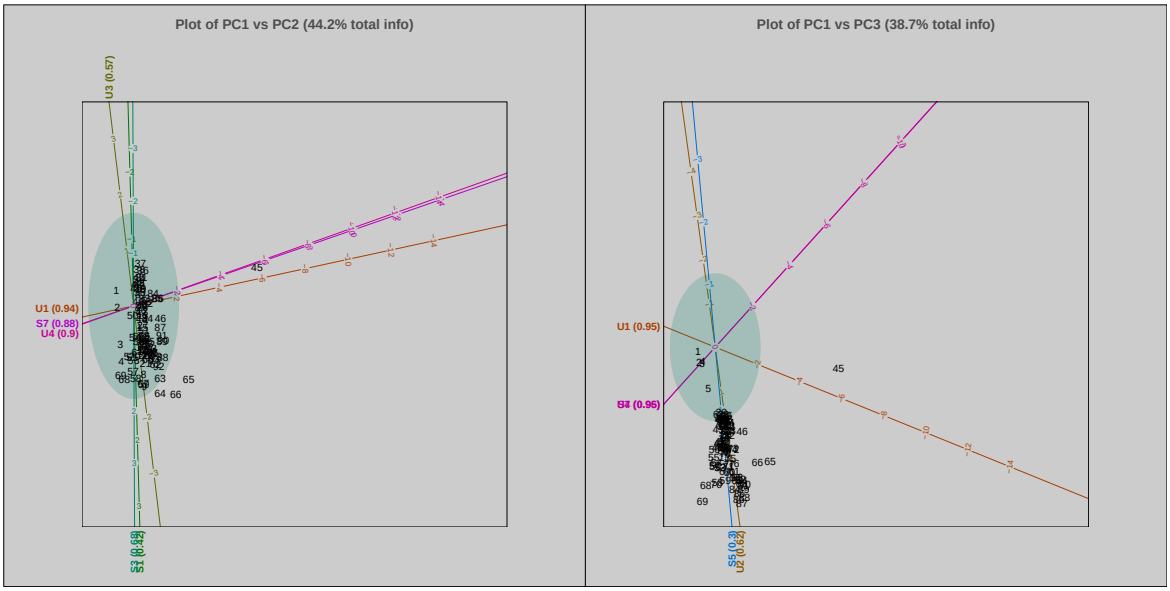
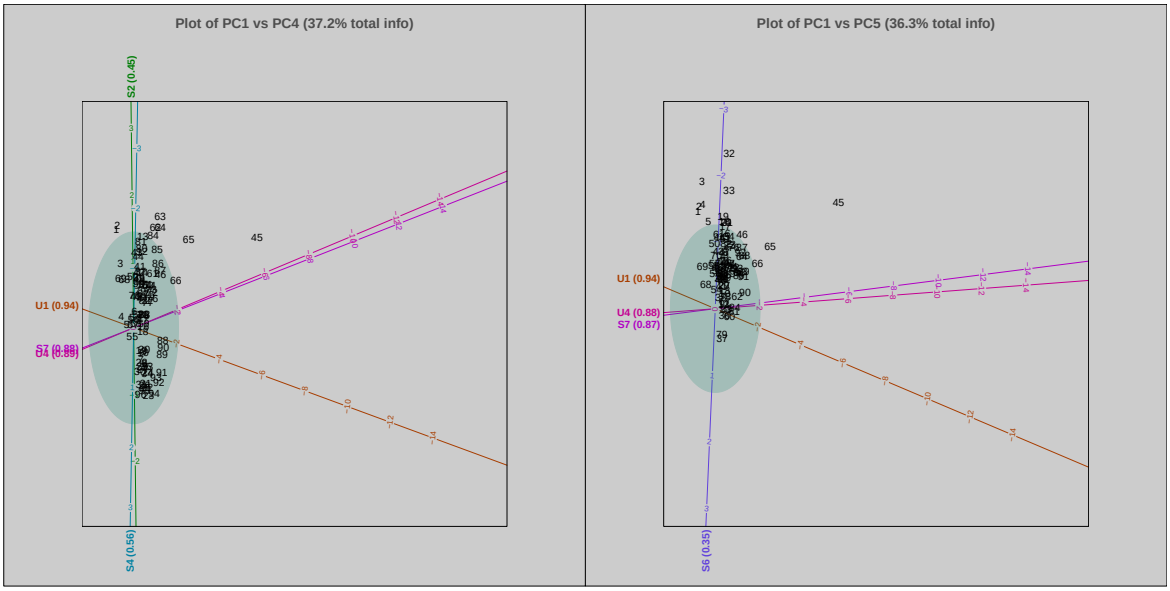


Figure 4.25: Variable contribution plot for GG26 at $t = 53$



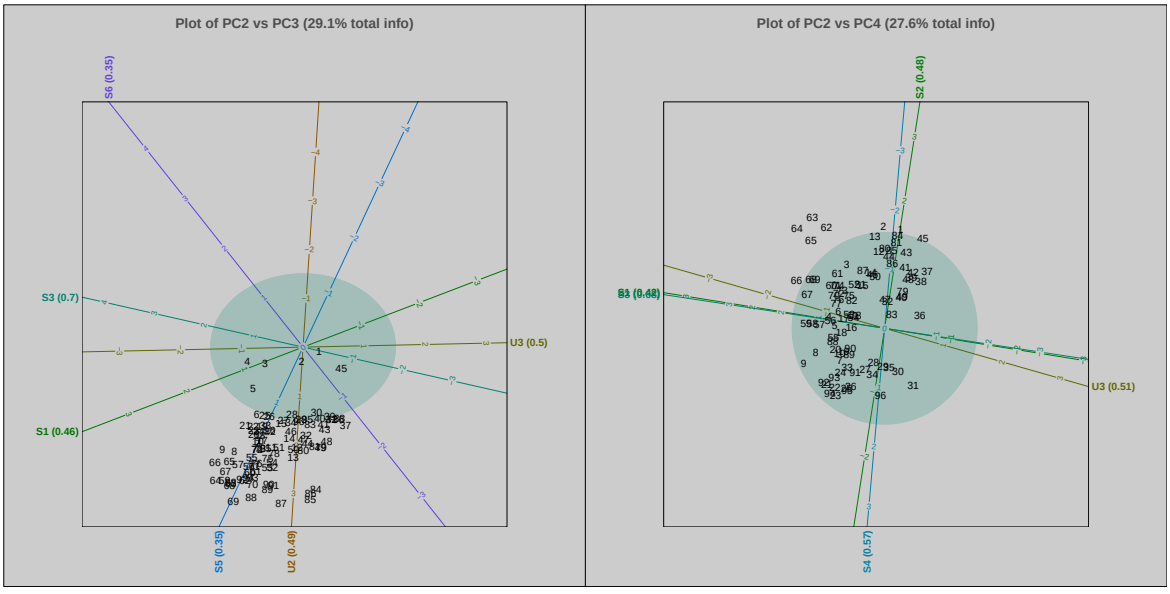
(a) PC1 vs PC2

(b) PC1 vs PC3



(c) PC1 vs PC4

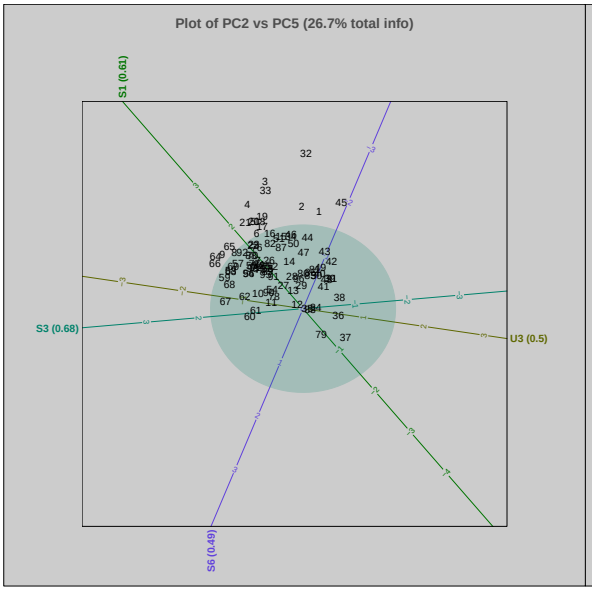
(d) PC1 vs PC5



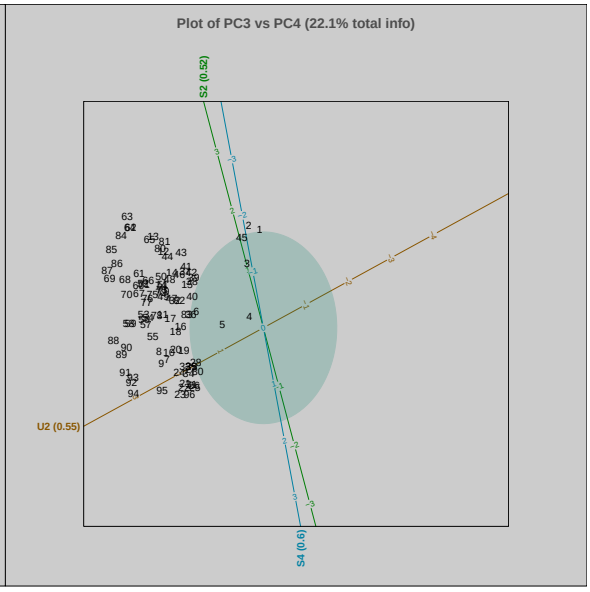
(e) PC2 vs PC3

(f) PC2 vs PC4

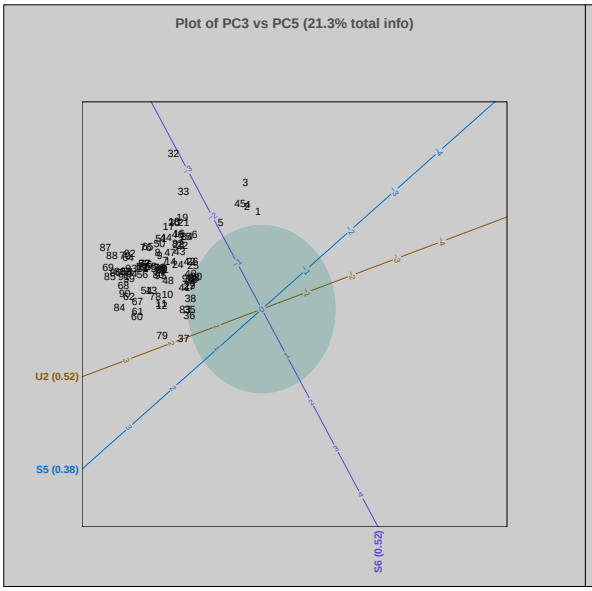
Figure 4.26: PCA plots of different principal component combinations for GG17



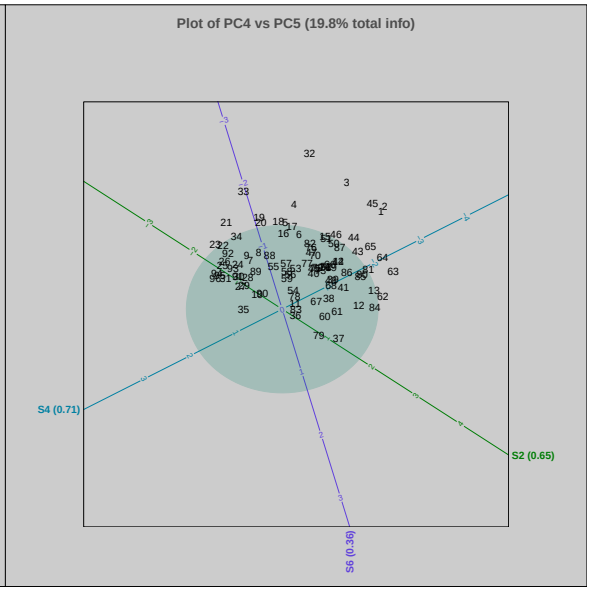
(g) PC2 vs PC5



(h) PC3 vs PC4

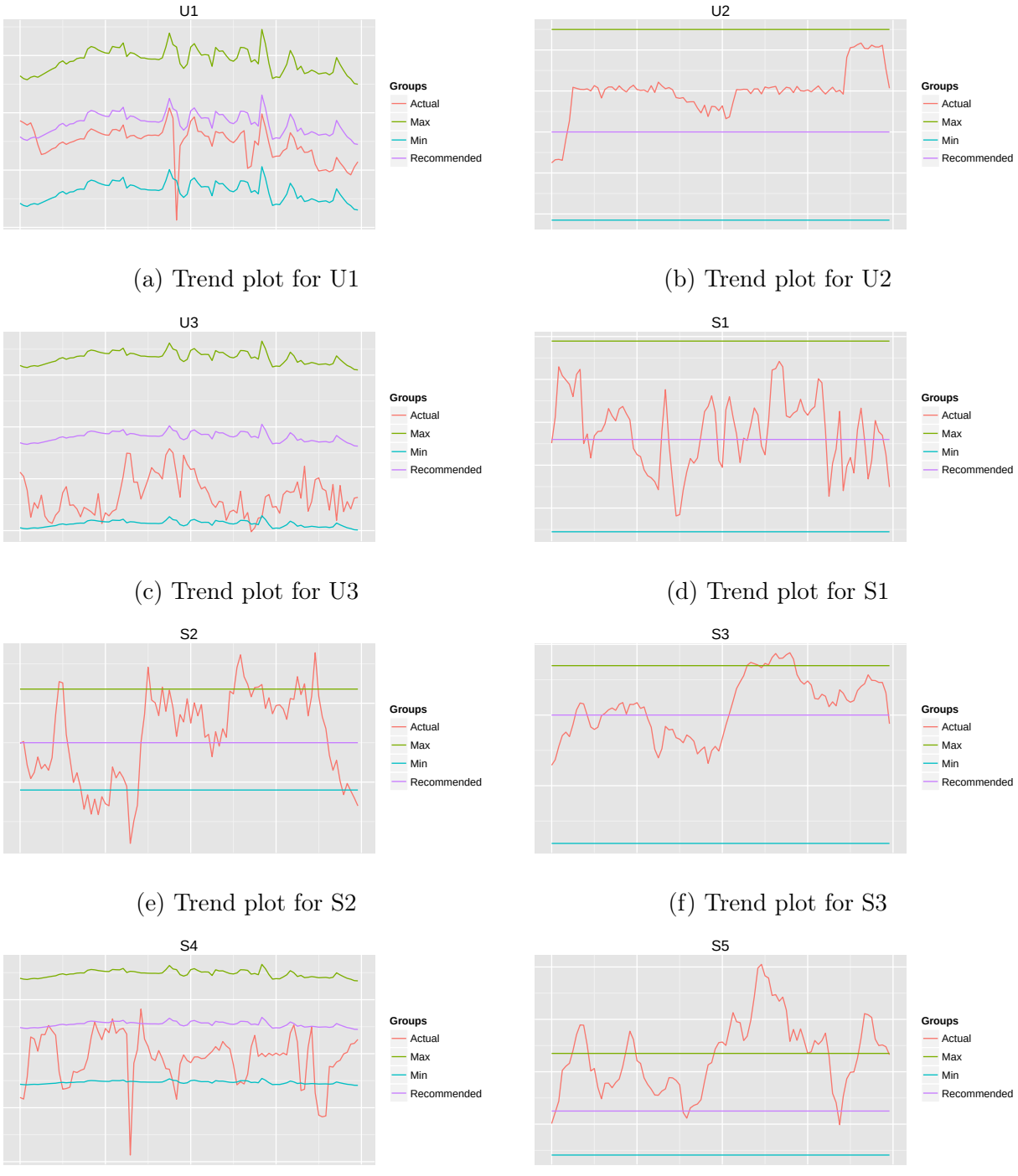


(i) PC3 vs PC5



(j) PC4 vs PC5

Figure 4.26: PCA plots of different principal component combinations for GG17 continued



(a) Trend plot for U1

(b) Trend plot for U2

(c) Trend plot for U3

(d) Trend plot for S1

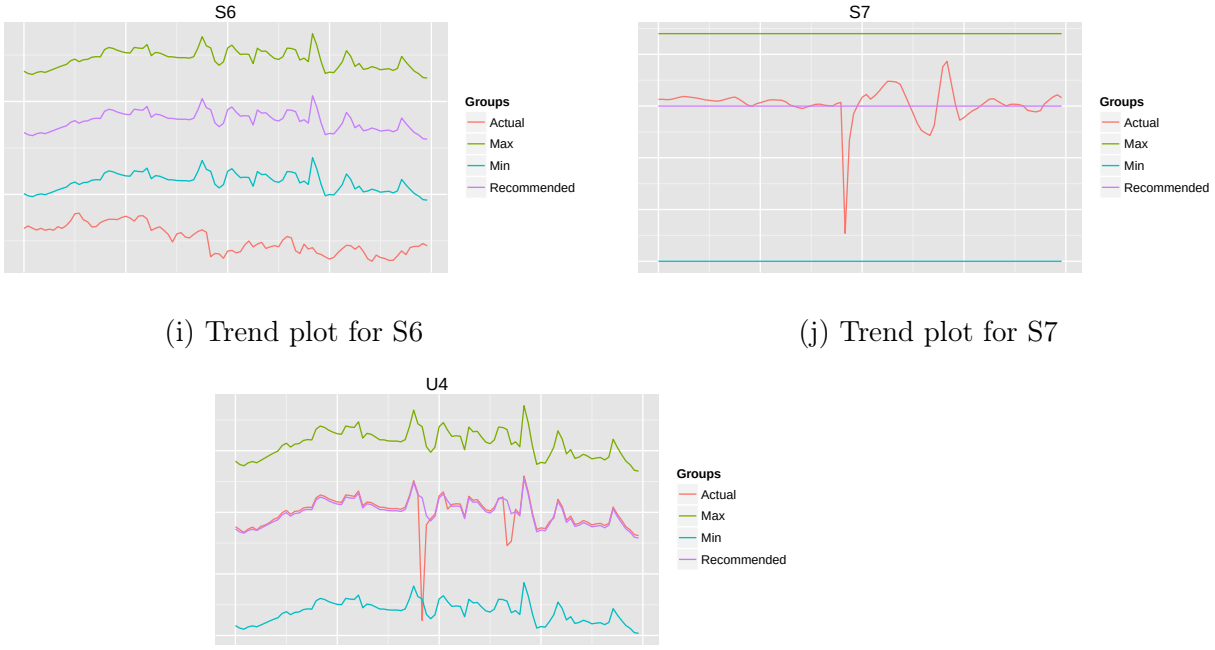
(e) Trend plot for S2

(f) Trend plot for S3

(g) Trend plot for S4

(h) Trend plot for S5

Figure 4.27: Trend plots of the process variables for GG17

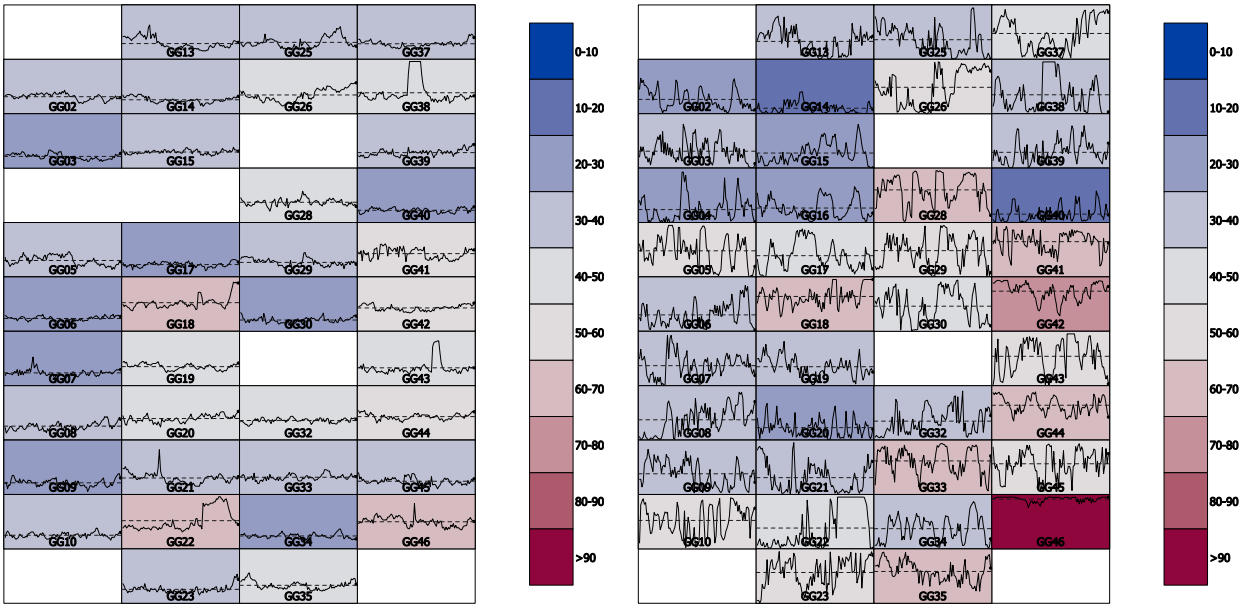


(i) Trend plot for S6

(j) Trend plot for S7

(k) Trend plot for U4

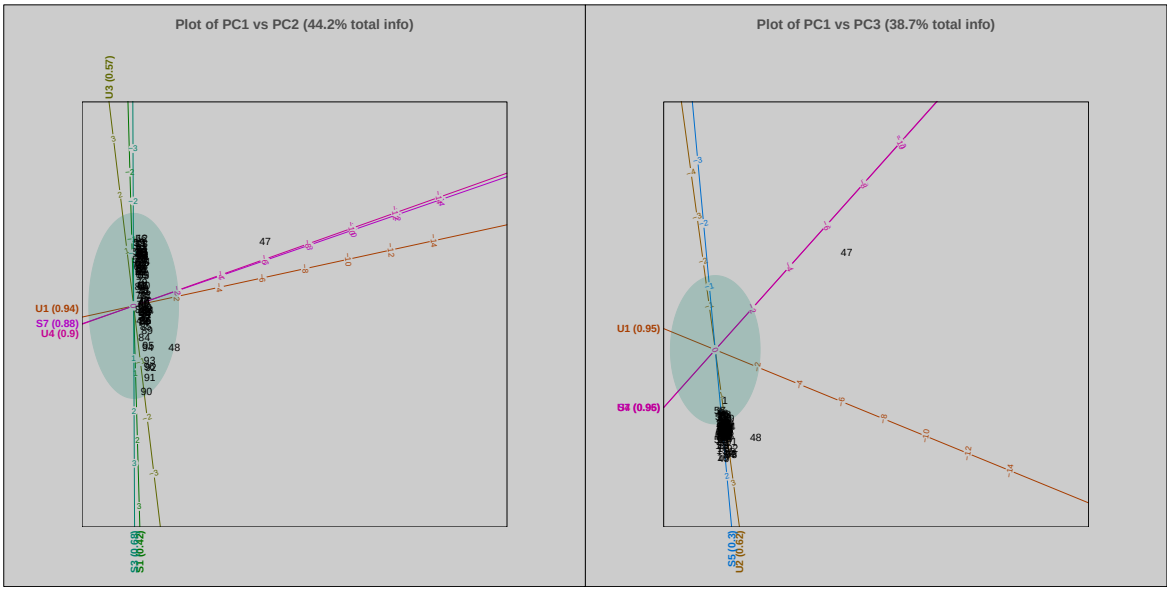
Figure 4.27: Trend plots of the process variables for GG17 continued



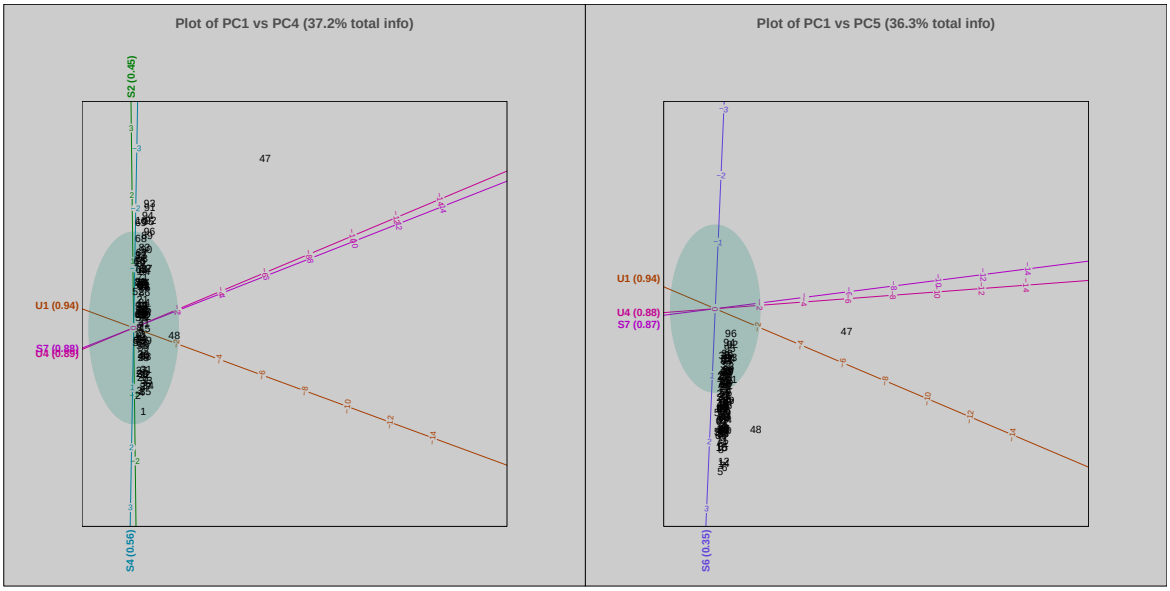
(a) Process driven GPI

(b) Integrated GPI

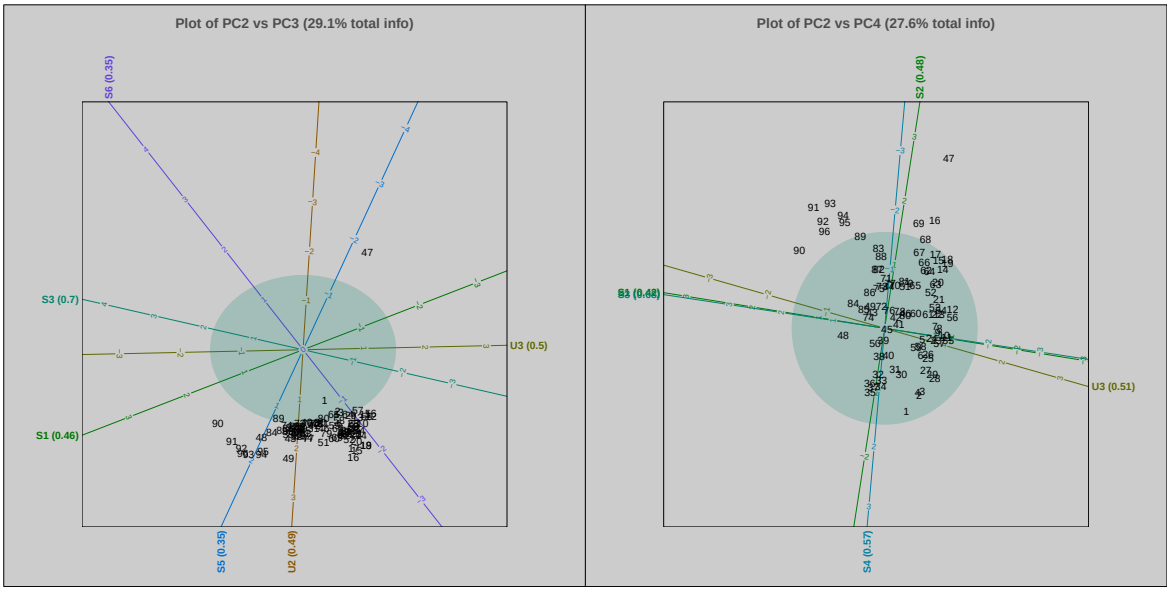
Figure 4.28: Comparison of fundamental and integrated GPI at low factory load values



(a) PC1 vs PC2 (b) PC1 vs PC3

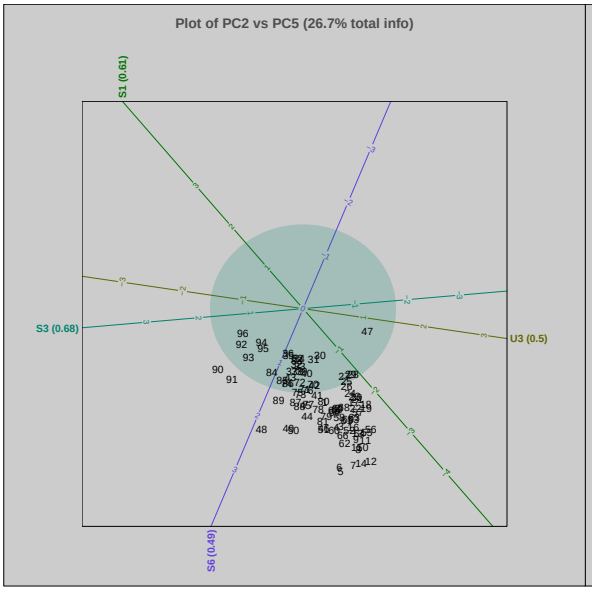


(c) PC1 vs PC4 (d) PC1 vs PC5

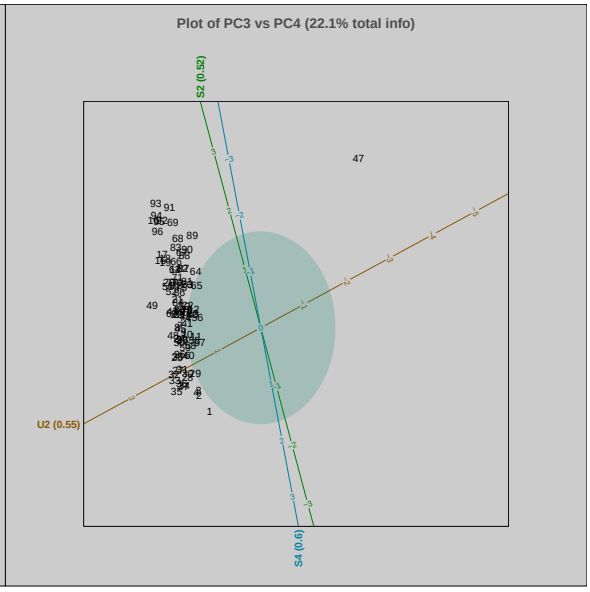


(e) PC2 vs PC3 (f) PC2 vs PC4

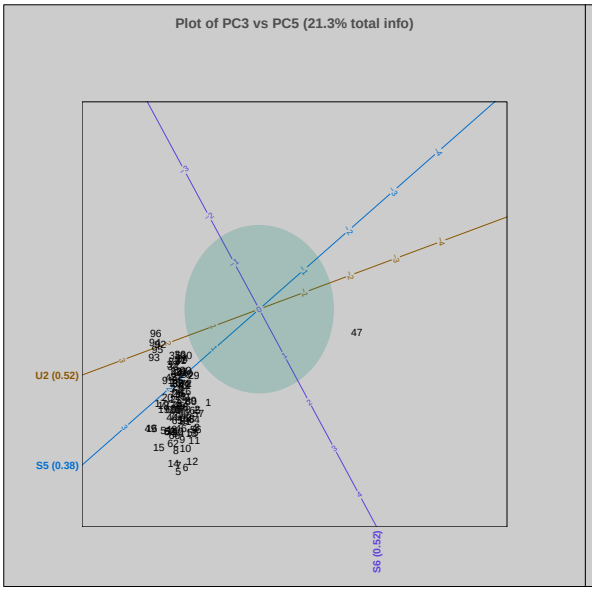
Figure 4.29: PCA plots of different principal component combinations for GG46



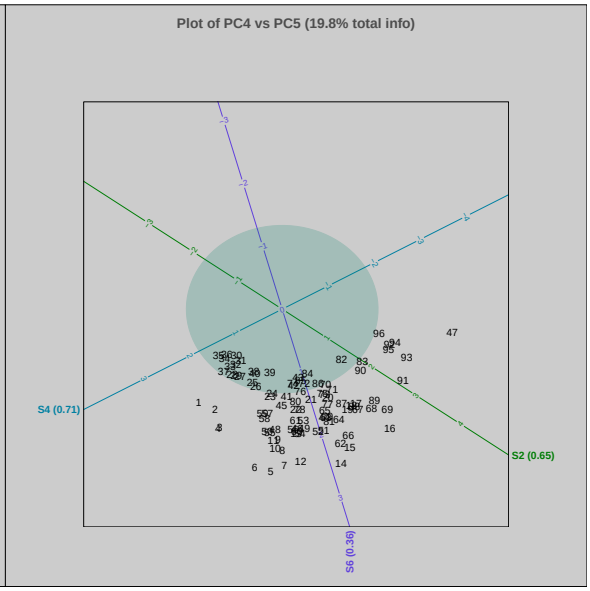
(g) PC2 vs PC5



(h) PC3 vs PC4

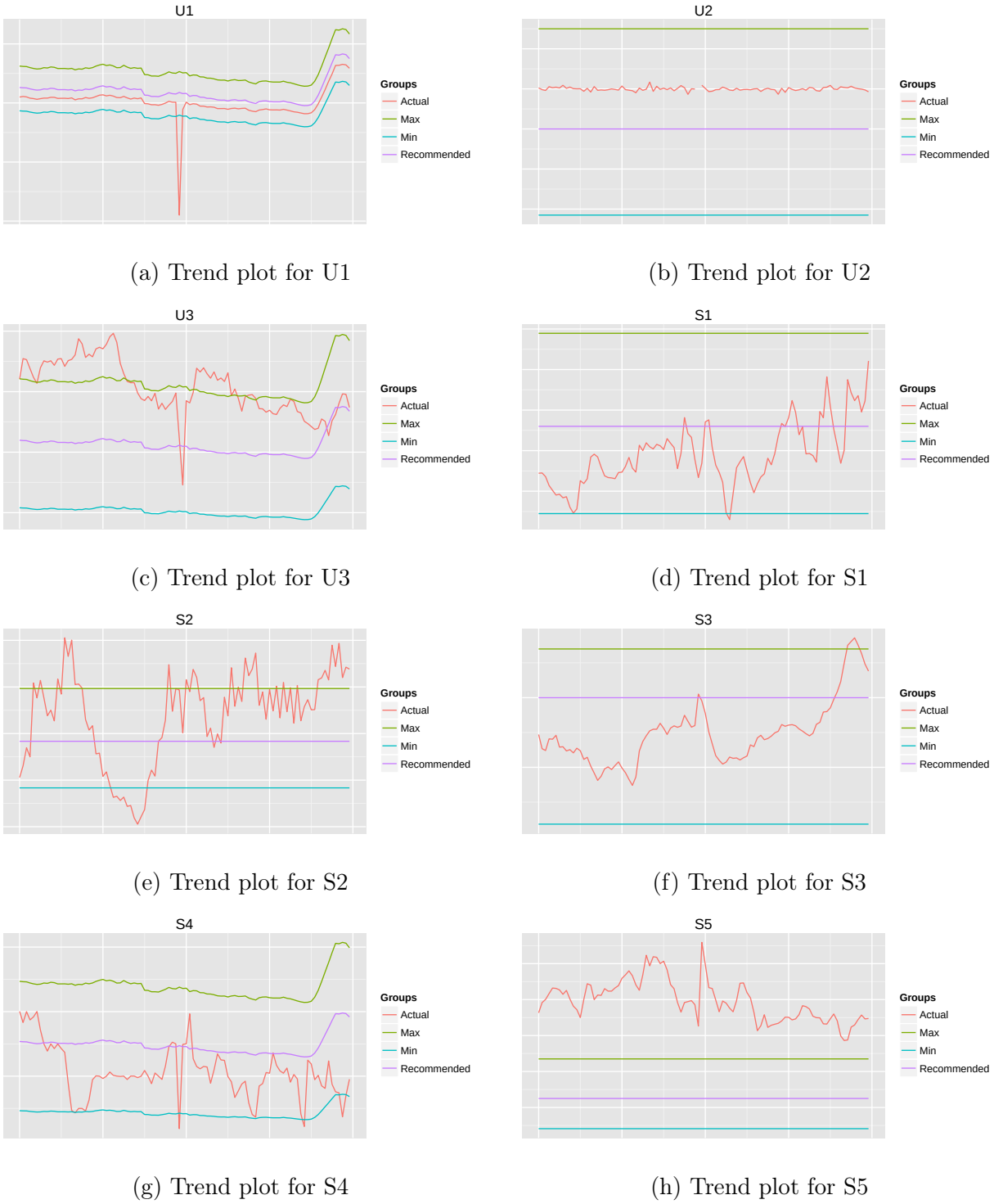


(i) PC3 vs PC5



(j) PC4 vs PC5

Figure 4.29: PCA plots of different principal component combinations for GG46 continued



(a) Trend plot for U1

(b) Trend plot for U2

(c) Trend plot for U3

(d) Trend plot for S1

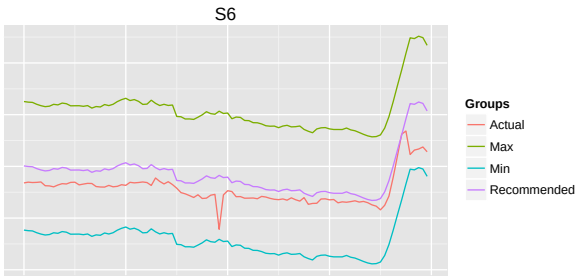
(e) Trend plot for S2

(f) Trend plot for S3

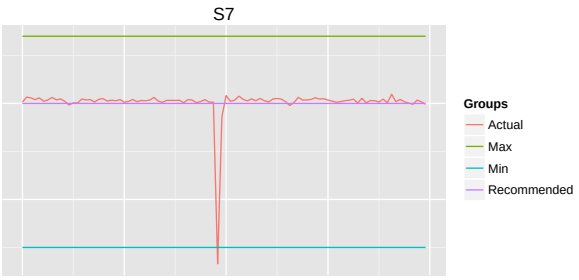
(g) Trend plot for S4

(h) Trend plot for S5

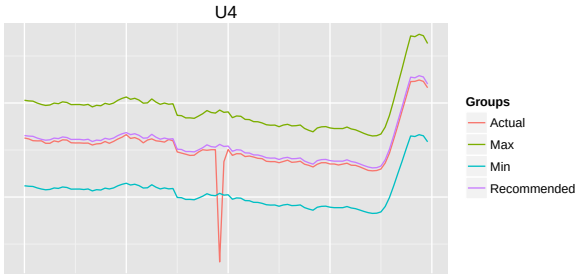
Figure 4.30: Trend plots of the process variables for GG46



(i) Trend plot for S6



(j) Trend plot for S7



(k) Trend plot for U4

Figure 4.30: Trend plots of the process variables for GG46 continued

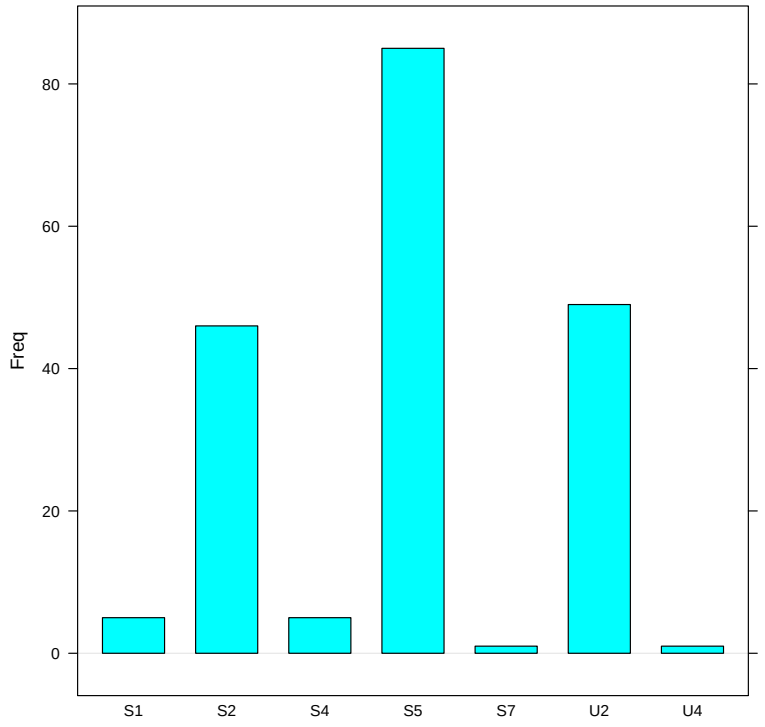


Figure 4.31: Frequency of variables as one of the two highest contributions to the T^2-value

4.3.5 CVA biplots

In this section the effect of applying CVA biplots to the new data (deviation from target (4.3.1)) will be investigated. CVA biplots are in general scale invariant as discussed in Gower *et al.* (2011) page 150, and the impact of the scaling is therefore expected to be small.

First, the maximum number of eigenvalues that needs to be included in the biplot analysis had to determine. The canonical variables will occupy at most $\min(J-1, p)$ dimensions of the canonical space. In this study there are 11 groups, and 11 variables, therefore the canonical variables can be fully represented in 10 dimensions. It can be observed from the scree plot in Figure 4.32 that the first 3 eigenvalues are larger than 1 which indicates that the first 3 eigenvalues should be retained. In Figure 4.33 the cumulative percentage variance explained is depicted. This is the variance explained in the original variables utilizing (3.4.10). The first 4 eigenvalues explains more than 80% of the total variance and indicates that the first 4 eigenvalues should be retained. Therefore, 4 eigenvalues were used in this study. These results are similar to the results obtained on the unscaled data in Section 3.4.4.

CVA biplots were constructed for all 6 combinations of 2 eigenvalues at a time from the 4 eigenvalues and the predictive measures in Section 3.4.3.1 calculated. In Table 4.8 the biplot quality in the original variables are provided. These results are again very similar to the results in Section 3.4.4 Table 3.12.

In Tables 4.9 the axis predictivity from (3.4.12) is provided. In Table 4.9 the axes with a minimum axis predictivity of 0.4 are highlighted. Comparing Table 4.9 with Table 3.13 in Section 3.4.4, it can be concluded that the predictivities are similar.

Table 4.8: CVA Biplot Quality in Original Variables

CVA Dimensions	Biplot Quality
1 and 2	62.00
1 and 3	46.16
1 and 4	49.73
2 and 3	32.88
2 and 4	36.45
3 and 4	20.62

The CVA biplots for all the different combinations are provided in Figure 4.34. Comparing these plots to Figure 3.24 in Section 3.4.4, it can be concluded that the results are virtually identical. Therefore, the scaling performed has a minimal impact on the CVA biplots, and it is concluded that the CVA biplots can be utilized for the monitoring of the differences between gasifiers on the scaled data as well.

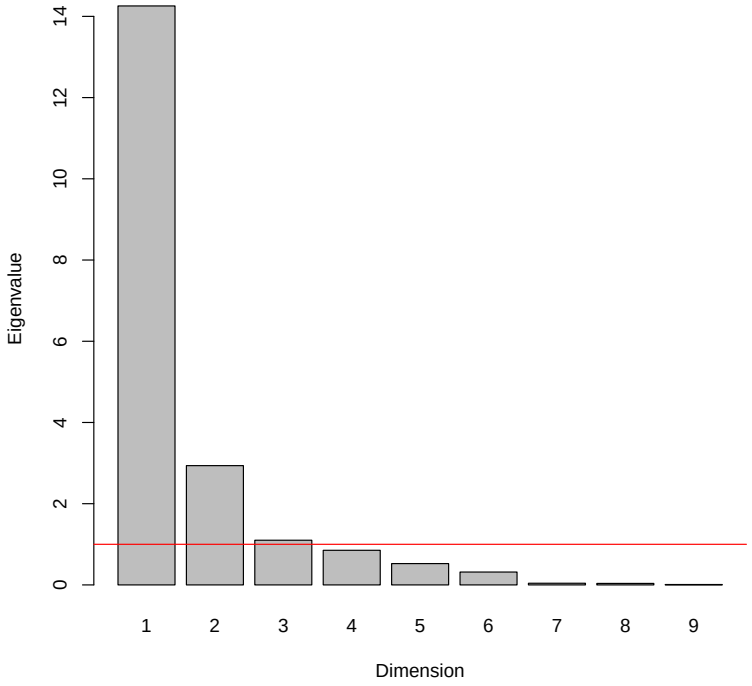


Figure 4.32: Scree Plot

Table 4.9: Axes Predictivity Values

Dimensions	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
1 and 2	0.28	0.04	0.98	0.54	0.01	0.61	0.34	0.77	0.73	0.16	0.39
1 and 3	0.49	0.03	0.96	0.50	0.24	0.10	0.36	0.03	0.74	0.19	0.55
1 and 4	0.34	0.47	0.96	0.39	0.04	0.12	0.85	0.00	0.50	0.26	0.30
2 and 3	0.35	0.07	0.02	0.28	0.23	0.51	0.05	0.80	0.47	0.19	0.43
2 and 4	0.20	0.51	0.02	0.17	0.03	0.52	0.54	0.77	0.23	0.25	0.19
3 and 4	0.42	0.50	0.00	0.13	0.25	0.02	0.56	0.03	0.24	0.29	0.35

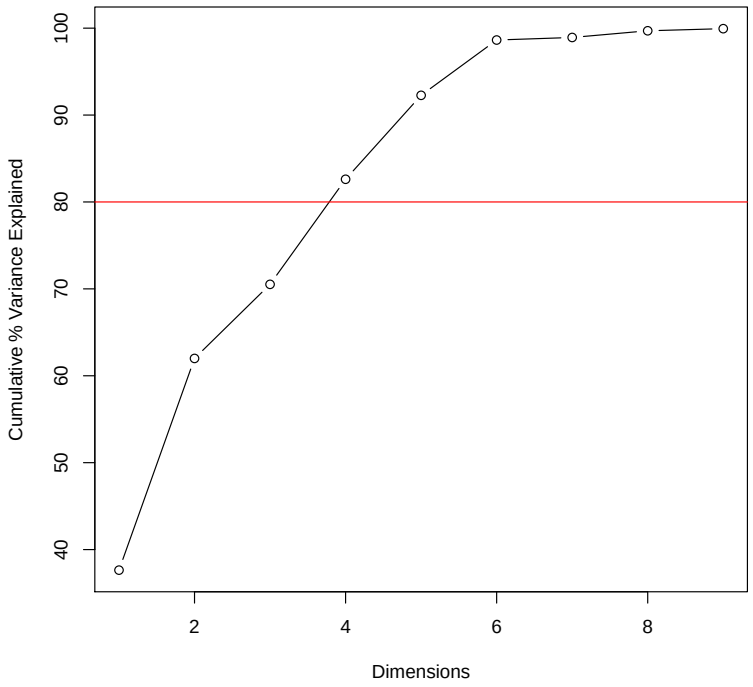
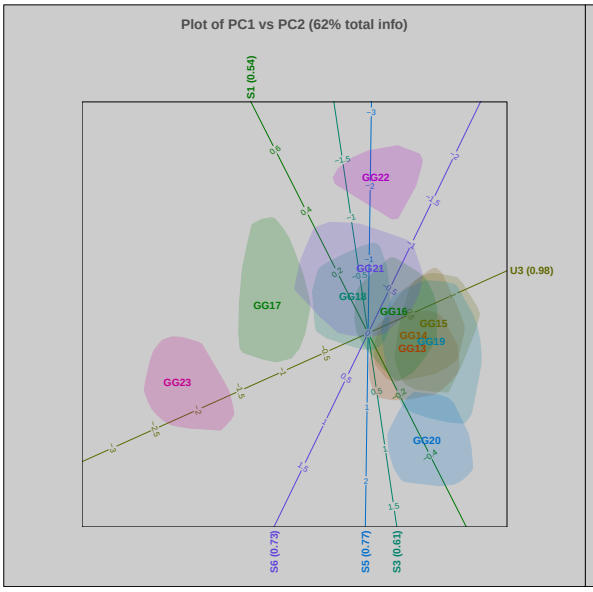
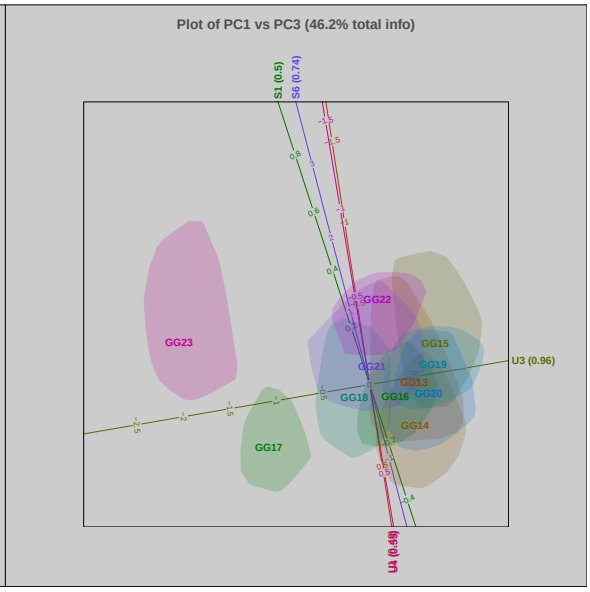


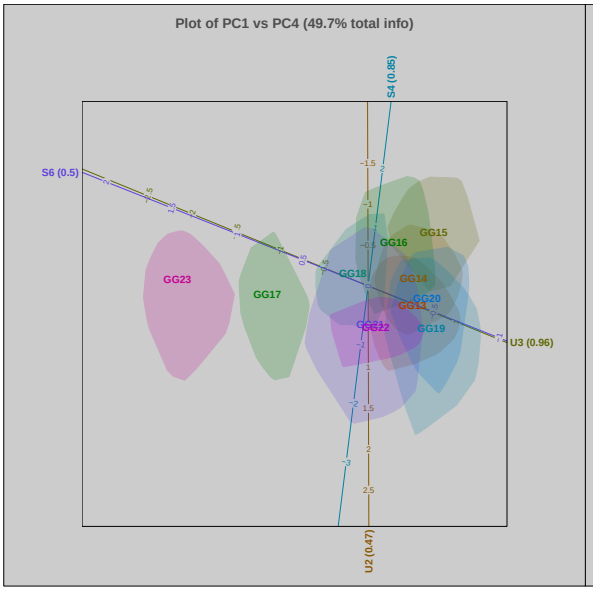
Figure 4.33: Cumulative Percentage Variance Explained



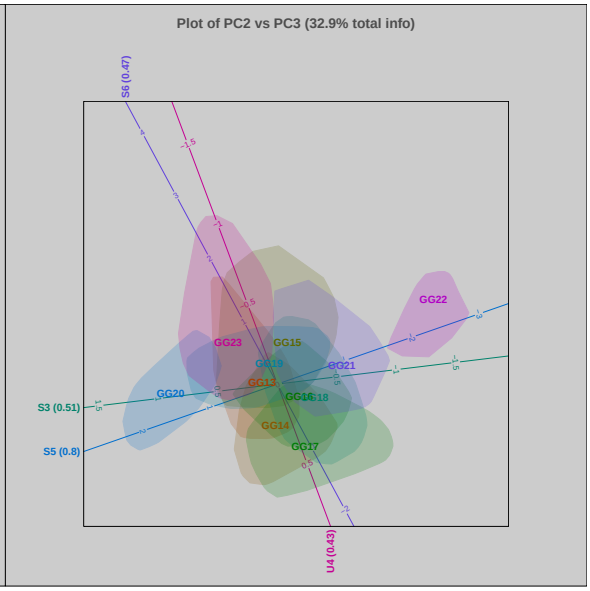
(a) CVA Biplot for PC1 Against PC2



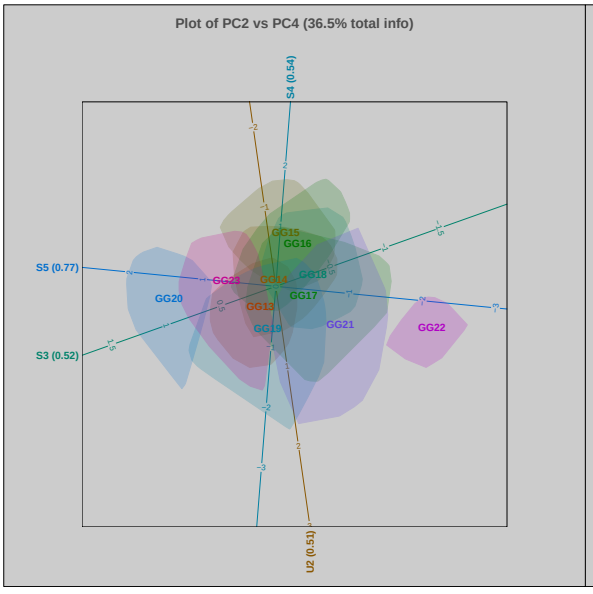
(b) CVA Biplot for PC1 Against PC3



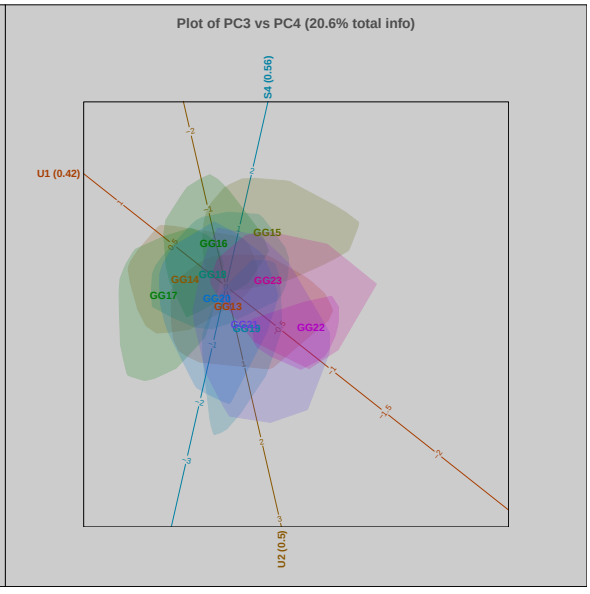
(c) CVA Biplot for PC1 Against PC4



(d) CVA Biplot for PC2 Against PC3



(e) CVA Biplot for PC2 Against PC4



(f) CVA Biplot for PC3 Against PC4

Figure 4.34: CVA biplots for the Combinations up to 4 Dimensions

4.4 Comparison of GPI indices

The different GPI indices have different advantages and disadvantages as discussed in the previous sections. The biggest disadvantage of the empirical GPI is the inability to adapt to different factory load settings (or more generally to any control measure). Therefore, the empirical GPI will not be discussed in this section as it is impractical for the current study under consideration. To compare the fundamental and integrated GPI, historical data will be used. A coincident table will be generated. The table is divided in four quadrants:

1. Both the integrated and fundamental GPI indicate that the process is performing within expectation.

2. The integrated GPI indicates that the process is performing within expectation and the fundamental GPI indicates that the process is deviating from expected performance.

3. The fundamental GPI indicates that the process is performing within expectation and the integrated GPI indicates that the process is deviating from expected performance.

4. Both the integrated and fundamental GPI indicates that the process is deviating from expected performance.

In this section a comparison of the integrated GPI to the fundamental GPI, which are encapsulated in quadrants 2 and 3 above, are of interest. Aggregated data (fifteen minutes) from one week for one train were captured to demonstrate and discuss the comparison between the two methodologies. The data set consist of 4928 rows (n). Each row consists of the 11 variables (p) discussed in the previous sections and a grouping variable indicating the gasifier. The time stamp for each row is available as well. Table 4.10 depicts the first six rows of the data set. The times were replaced with index values. The data are scaled and centred using the mean and standard deviation of the reference set. It is therefore possible to compare the variables as they are all on the same scale. Therefore, it can be concluded from Table 4.10 that variable U2 was high on average for this time period, and variable S6 was very low on average for the same period.

Both the fundamental and integrated GPI values were calculated for the historical dataset. A confidence (α) value of 90% was chosen for the integrated GPI i.e., any value above 90% will be classified as deviating from expected performance. This value is in line with best practises from the multivariate monitoring literature (MacGregor and Kourti, 1995; Russell *et al.*, 2000). Employing the integrated GPI, 1224 values from the total of 4928 were classified as deviating from expected performance. Using an index value of 90% for the fundamental GPI only 155 values were classified as deviating from expected performance. Table 4.11 contains the populated coincident values. There are

no values in quadrant 2, and 1069 values in quadrant 3. This indicates a large discrepancy between the classification by the integrated GPI versus the fundamental GPI. Three possible causes for this discrepancy are:

1. The confidence value for the fundamental GPI is too high.
2. The integrated GPI classifies values as deviating from expected performance that is in fact performing as expected. This can be due to a reference set that is not representative of the current operating region.
3. The fundamental GPI is not sensitive enough to specific deviations.

Referring to Equation (4.1.9) the fundamental GPI is defined as a weighted offset from the target values. Therefore, choosing a confidence value of 90% is subjective as the value does not have any fundamental meaning. A bigger concern is that it is theoretically possible that only one variable deviate from the target value while the remaining variables are very close to their target value. This can yield a relatively low index value even though a large performance deviation is taking place. The basic premise behind an index value is to guide a user to the performance deviations, and it is therefore important that deviations are detected correctly. However, too many false positives will lead to information overload. Values with both an integrated GPI value of higher than 90% and a fundamental GPI value of lower than 50% occurred 147 times. These values represent the extreme case of the discrepancy between the fundamental and GPI values, and were investigated in more detail.

The goal of the analysis is to determine if these values are false positives i.e., the integrated GPI classified these values as deviations from expected performance while they are actually performing as expected. The variable contributions to the T^2-values (see (4.2.14)) were calculated for each of the 147 values. The data for GG01 contained 64 out of the 147 values with a fundamental GPI value below 50% and integrated GPI value above 90%. Figures 4.35 to 4.40 depicts the actual values of some of the process variables as the black solid line over this period for GG01. The lower, upper and target $(dopt - dopt_{min}, dopt, dopt + dopt_{max})$ values are depicted by the dashed blue lines. The red dots indicates a time t where the specific variable was one of the two largest contributing variables to the T^2-value. Note that the red dots only indicate values where the fundamental GPI was smaller than 50% and the integrated GPI was larger than 90%. It is therefore expected that a large number of deviations from expected performance will not be highlighted. The goal of this analysis is to investigate these specific cases of discrepancy only. The performance of the individual performance indices was discussed in previous sections. Variables U2, S1, S2, S3, S4 and S6 were each indicated as one of the two largest contributing variables at least once. Each plot will be discussed briefly:

- As discussed previously, variable U2 is a control variable with very low variability. In Table 4.12 the mean and standard deviation of the scaled data ($\boldsymbol{Z}$) are tabulated. The mean for variable U2 (0.05) indicates that U2 was operated very close to the recommended target value. Also the standard deviation value of 0.2 is relatively small and indicates that the variable was well controlled. Therefore, variable U2 will exhibit low variability in the reference data set $\boldsymbol{Z}$, and small changes will result in a high T^2-value and corresponding threshold value. It can be observed from Figure 4.35 that various step changes occurred and were flagged by the integrated GPI. It were therefore concluded that these values are indeed deviations from expected performance, and should be highlighted. Therefore, the confidence value of the integrated GPI gives more accurate information compared to the results of the fundamental index.

- Variable S1 was operated below the target value during the reference period (see Table 4.12). A few of the values marked as deviations were however clearly not large deviations as they are very close to the target values (see Figure 4.36). On closer inspection it was realised that variable S1 was always the second highest contributing variable with relatively low values when it occurred. Similarly the contribution to the T^2-value by variable S3 was less than 15% (see Figure 4.38) when it occurred. These two variables can therefore be ignored. It would be prudent to add an additional filter to the relative size of the second largest contributing variable in future studies.

- The values indicated as deviations on Figures 4.37 and 4.39 clearly deviate from expected performance for variables S2 and S4.

- Variable S6 is the largest contributing variable to the T^2-values in 47 of the 64 cases for GG01. It can be observed from 4.39 that variable S6 was very low over the whole period. It can be observed from Table 4.12 that S6 is below the recommended target value in general. The values in this specific period under investigation were -2.5 on average, and were clearly deviating from expected performance.

Therefore, it can be concluded that the values classified as deviations from expected performance with the integrated GPI were not false positives. The integrated GPI flags periods of performance deviations which were misclassified by the fundamental GPI.

Table 4.10: Input data for GPI indices

DateTime	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4	GG
1	-0.88	1.80	1.05	-0.51	0.43	-1.15	-0.02	-0.03	-2.96	0.45	0.06	01
2	-0.89	1.80	0.27	-0.49	-0.17	-0.98	-0.16	0.01	-3.04	0.18	0.06	01
3	-0.88	1.80	0.45	0.14	0.21	-1.04	0.24	0.03	-3.06	0.08	0.07	01
4	-0.88	1.80	0.49	0.34	-0.39	-1.01	0.08	0.06	-2.84	0.09	0.07	01
5	-0.88	1.80	0.47	0.40	0.25	-1.02	0.31	0.09	-2.77	0.10	0.07	01
6	-0.89	1.80	0.60	-0.12	0.85	-1.08	0.17	0.17	-2.93	0.11	0.06	01

Table 4.11: Coincident table results for one week for process performing within expectation for the integrated versus fundamental GPI

		Fundamental	
		Yes	No
Integrated	Yes	3704	0
	No	1069	155

Table 4.12: Mean and standard deviation values for scaled reference set $\boldsymbol{Z}$

	U1	U2	U3	S1	S2	S3	S4	S5	S6	S7	U4
Mean	-0.08	0.05	0.44	-0.27	-0.54	0.42	-0.19	0.33	-0.73	-0.03	-0.04
SD	0.28	0.20	0.50	0.33	0.68	0.61	0.49	0.80	0.51	0.14	0.20

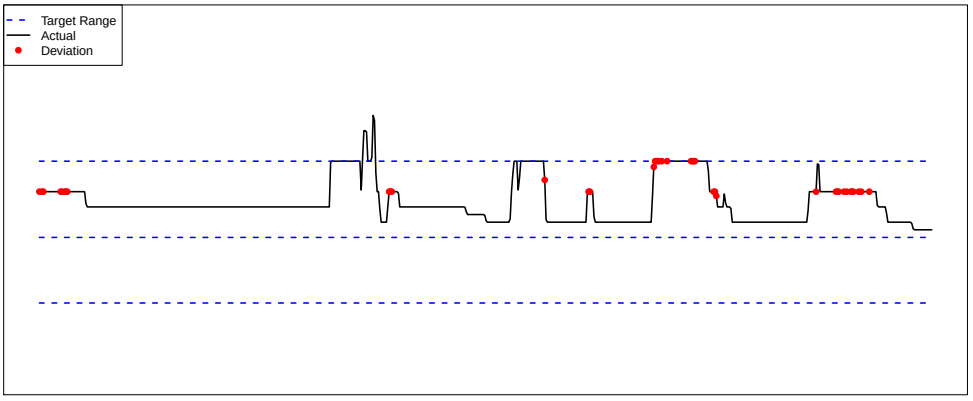


Figure 4.35: Trend plots for GG01 variable U2 highlighting the values with large discrepancies between the fundamental and integrated GPI

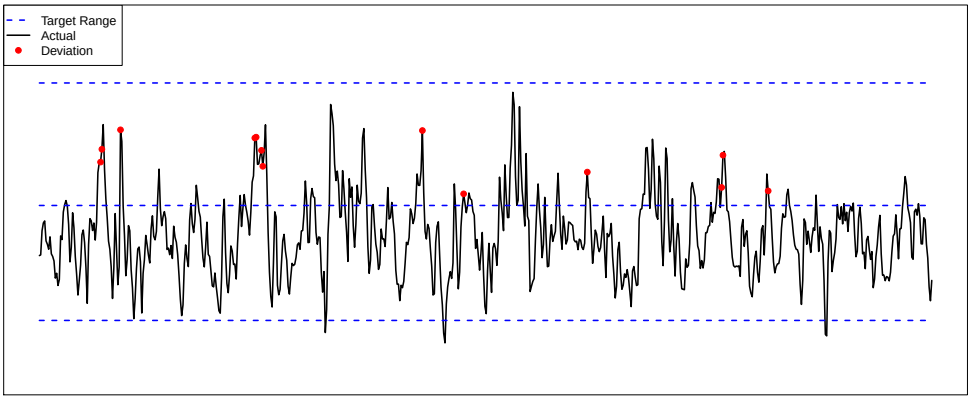


Figure 4.36: Trend plots for GG01 variable S1 highlighting the values with large discrepancies between the fundamental and integrated GPI

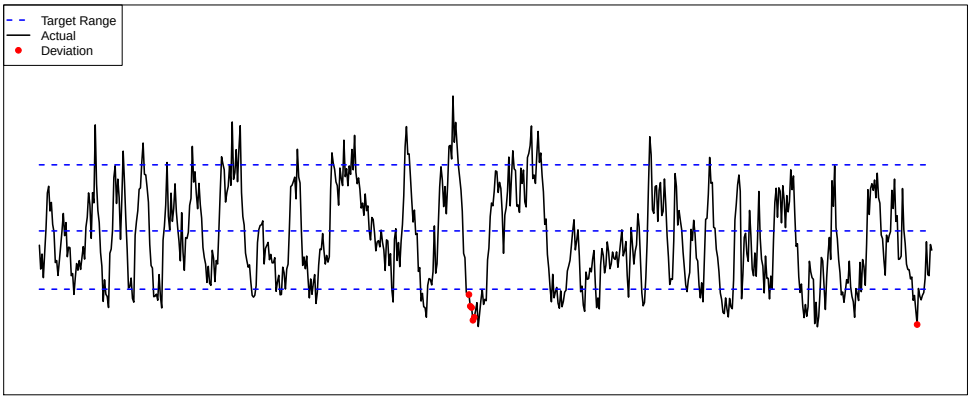


Figure 4.37: Trend plots for GG01 variable S2 highlighting the values with large discrepancies between the fundamental and integrated GPI

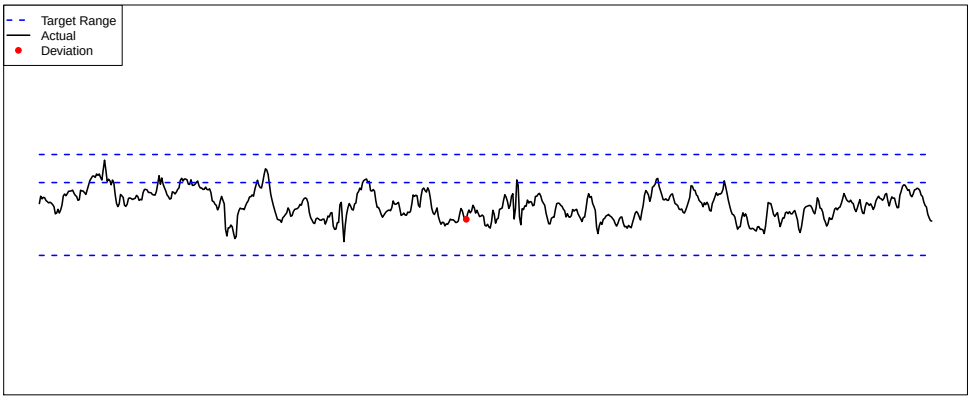


Figure 4.38: Trend plots for GG01 variable S3 highlighting the values with large discrepancies between the fundamental and integrated GPI

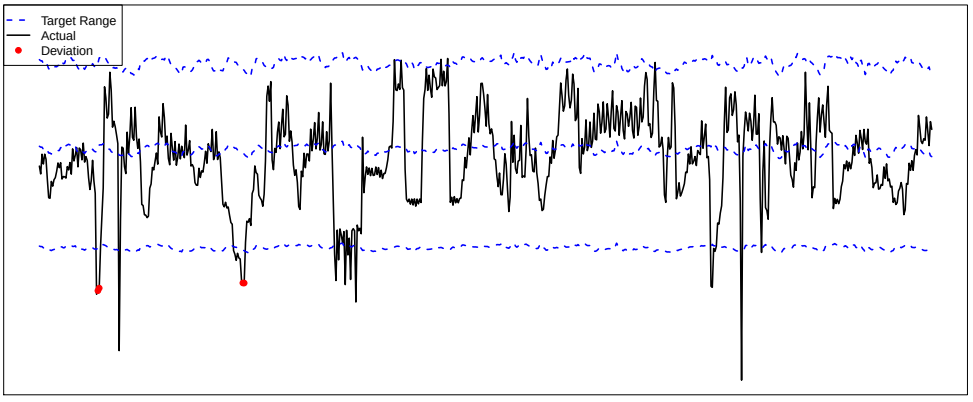


Figure 4.39: Trend plots for GG01 variable S4 highlighting the values with large discrepancies between the fundamental and integrated GPI

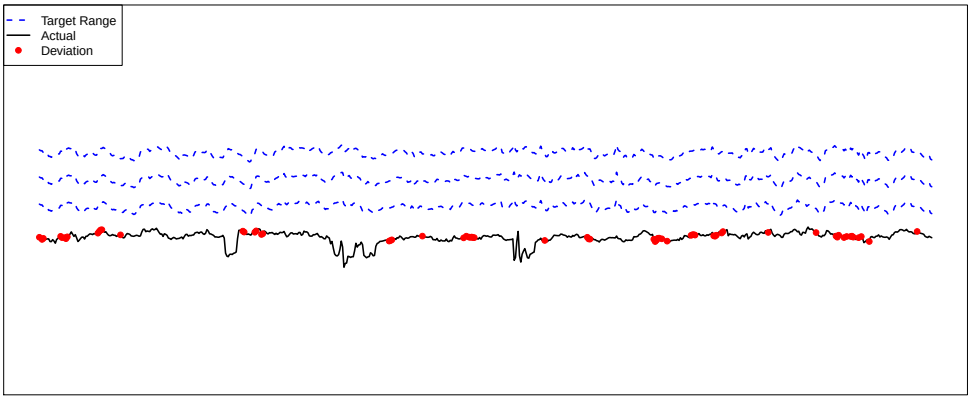


Figure 4.40: Trend plots for GG01 variable S6 highlighting the values with large discrepancies between the fundamental and integrated GPI

4.5 Conclusions

In this chapter three different approaches to a gasifier performance index (GPI) were investigated and compared. The different indices are:

- *Fundamental GPI* - a purely process driven performance index.
- *Empirical GPI* - a purely data driven performance index.
- *Integrated GPI* - an integrated process and data driven performance index.

In Section 4.1 the fundamental (process driven) GPI was discussed and implemented. The fundamental GPI is based purely on process knowledge and is implemented as a weighted sum of the variables, adjusted by the correction factors and scaled to the same range. The correction factors are related to a factory load variable. The fundamental GPI has several advantageous characteristics. Some of the advantages are:

1. There is no need to define a reference set.
2. The GPI calculation is computationally efficient.
3. The contribution of each variable to the overall GPI is easy to calculate.
4. The GPI calculation is intuitive and easy to relate to the actual production process. This leads to ease of acceptance by the production engineers.

The fundamental approach however possesses two big disadvantages:

1. The GPI value is subjective as all the underlying input values are supplied by subject matter experts.
2. The GPI is by definition univariate, and does not take into account any multivariate relationships between the variables.

The fundamental approach can be generalised to any process given that the required information is available.

In Section 4.2 a purely data driven (empirical) approach to the GPI was developed and demonstrated. This index gives comparable results to the fundamental GPI values. The methodology is formalised in Figure 4.12. This methodology is proposed as a general data driven performance index as it is objective, and very little prior knowledge of the system is required. Following all the steps in the methodology will lead the user in understanding the underlying process and will provide the interrelationships among the different variables. There are however some disadvantages to this methodology:

- All the variables are equally weighted. This is not necessarily a disadvantage, but the user should be aware of this property.
- Both high and low values are equally weighted for the variables. This is again not necessarily a disadvantage, but may not be optimal.
- A reference data set is required. There is however some advantage in going through the process of obtaining the reference set as valuable knowledge of the underlying process is gathered.
- The empirical GPI is more computationally intensive than the fundamental GPI. However, the mathematical calculations can be stored and do not need to be calculated in real time.

Finally in Section 4.3 an integrated GPI was proposed and demonstrated. The integrated GPI retains advantages from both the fundamental and empirical GPI, and eliminates some of the major disadvantages.

- As for the fundamental GPI, the integrated GPI allows for the dynamic adjustment of the recommended value for each variable subject to changes in a control parameter (factory load in the case of the GPI). Also, it allows for the weighting of variables above or below the recommended value. This is very important, especially for temperature and pressure variables. Generally lower temperatures and pressures can lead to non optimal performance, but high temperatures and pressures can lead to a safety risk. It is therefore important to penalize an index value more for a higher value for these variable types, than for a identical offset on the lower side.
- Similar to the empirical GPI, the integrated GPI is objective (except for the choice of variables to include) and the final weighting of the variables are obtained from the underlying algebraic properties of the PCA and T^2-statistics. Additionally, the process of obtaining the reference set, number of principal components to include as well as the axis predictivities lead to valuable insight into the underlying process. However, the most important advantage of the data driven approach is that it is takes into account the multidimensional character of the gasification process.

The following integrated GPI methodology is proposed:

1. For a given reference set $\boldsymbol{X}$), calculate the scaled offset from the recommended target values using the adjusted equation (4.3.1). This yields a data set $\boldsymbol{Z}$ of offsets from the recommended values scaled by the appropriately specified delta values for the minimum and maximum values. Additionally the correction factor for the factor load variable is applied. Then continue with the empirical GPI algorithm as depicted in Figure

4.12. Note the same scaling is performed on the new observational vectors $\boldsymbol{x}$ to yield $\boldsymbol{z}$ before performing the calculations.

The integrated GPI implementation was demonstrated for the Eastern factory. It was noted that the reference set period and train remained the same as for the data driven approach, and it was demonstrated that the underlying process knowledge was still present and captured in the reference set. The axis predictivity results lead to similar results than the empirical approach, with the exception of one extra principal component retained by the permutation test. The extra principal component was found to represent the one variable (S6) that was not well represented in the purely data driven approach. It can be argued that the scaling step removed some of the superfluous variance in the data and therefore improved the representation of the remaining variables. This data structure discovery process is one of the major advantageous of applying a data driven approach as part of a scientific discovery process.

The integrated GPI methodology was implemented and compared to the fundamental GPI in Section 4.4. It was concluded that the performance of the integrated GPI was superior to the fundamental GPI. The empirical GPI was not included in the comparison as it could not be used under low factory load conditions. This is a major disadvantage and excludes it for practical implementation.

These results can be generalized to any industrial process as follows:

- If the process knowledge is available:

    1. Implement and test the integrated GPI in parallel with the fundamental GPI to gain confidence from the plant engineers. Deviations between the two approaches can lead to insight in the advantageous of the integrated approach. The recommended values and the control variable effect should be monitored and adjusted periodically by utilizing the information from the data driven approach. Additionally the reference set should be reevaluated periodically to ensure the operating conditions have not changed.

- If the process knowledge is not available:

    1. Implement the empirical approach as it has many inherent advantageous and will lead to an in depth understanding of the process and the data. If control variables exist use the data driven approach to find the underlying effect of the control variables on the dependent variables. Use the data to capture the normal operating range and the delta values for the minimum and maximum values.
    2. Utilising the knowledge gained from the empirical approach, implement the integrated approach (and possibly for validation the fundamental approach) in parallel and improve until the integrated

approach performance meet the requirements. This will not only lead to a better performance index, but also the capturing of the process knowledge that was not available initially.

In conclusion, a performance index visualized on an appropriate and interactive graph is invaluable in the monitoring of multiple similar production processes, as it makes it easy to visually identify production processes not performing as expected. These processes can then be analyzed in detail using the proposed multivariate biplots. The process of developing and implementing these performance indices and multivariate methods are invaluable as part of the scientific discovery.

In summary, in this chapter a new approach to process deviation monitoring on many variables was presented based on the confidence (α) value at a specified T^2-value. This developed and presented methodology was proposed as a general data driven performance index as it is objective, and very little prior knowledge of the system is required. A novel multivariate gasifier performance index (GPI) was developed, which integrates subject matter knowledge with a data driven approach for real time performance monitoring.

In the next chapter the software implementation and design choices for the MSPEM™ application will be discussed. This will include the underlying infrastructure, the data capturing and storage facilities, statistical calculation procedures and the visualisation tools. A demonstration of the MSPEM™ application related to this study will also be provided.

Chapter 5

Software Infrastructure

Chapters 1 to 4 focused on the statistical methodologies implemented and employed for the real-time multivariate application. In this chapter the software development required to implement the real-time multivariate application will be presented. The choice of software infrastructure has far reaching consequences on the complete process monitoring application. It starts with the interface to the plant DCS systems and ends with the user interface presented in the user's browser. For the software infrastructure some basic criteria were set from the beginning:

- *Flexibility* - The software used for a specific section of the MSPEM™ package should be flexible i.e., the software should never be the constraint if new functionality must be added.
- *Scalability* - The software should be able to scale with the demand i.e., the number of users, or size of the database should not require an infrastructure change.
- *Fit for purpose* - Each module of the MSPEM™ package should be developed with a software that is ideally suited to the specific requirement. Therefore software should be chosen to be optimal for a specific module.
- *Independence* - Each module of the MSPEM™ package should be able to function independently i.e., if any part of the system is substituted for a different software product, it should have no impact on the rest of the system. Additionally if only one part of the system is needed it should be a fully functional complete product.

A graphical overview of the MSPEM™ software infrastructure as currently implemented is provided in Figure 5.1. In the remainder of this chapter each portion of the infrastructure will be discussed in detail.

Various R packages were developed as part of the MSPEM™ product. Table 5.1 lists the packages, a short description, the section where the package will be discussed, and the section where the source code is listed. The source

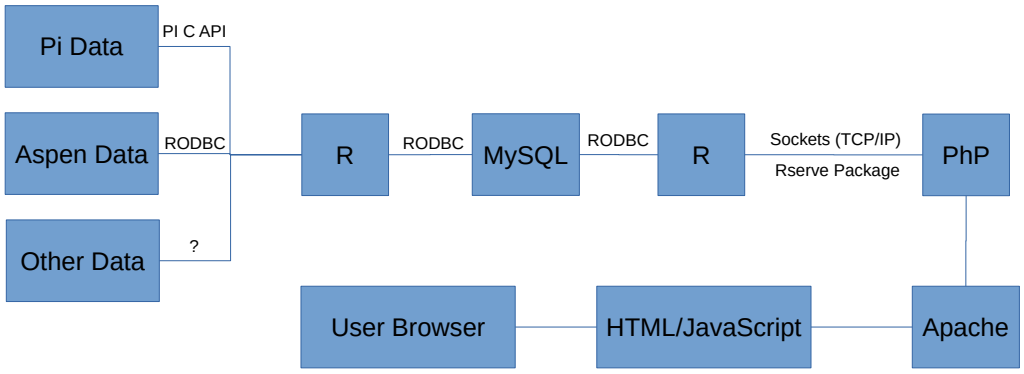


Figure 5.1: MSPEM™ Software Infrastructure Overview

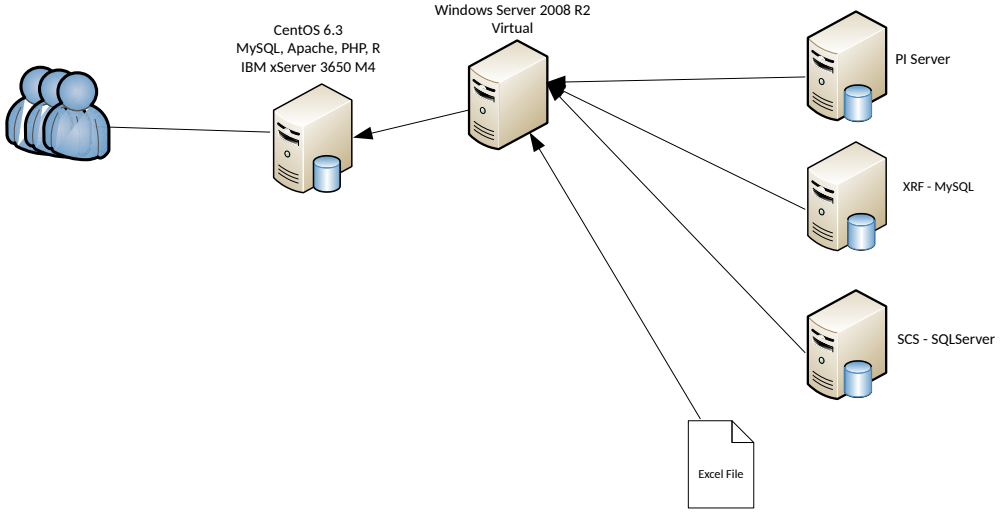


Figure 5.2: MSPEM™ Architecture Overview

code for the packages `sslgpicvc` and `sslscs` will not be provided due to confidentiality restrictions. However, these two packages contain the code specific to the facilities only, and the relevant data interface and statistical calculations are provided in separate packages. As the software packages are interrelated some aspects of a package could occur in sections prior to the discussion of the package. However, this will not confound the discussion of the specific section as the discussion will be largely focused on specific problems and solutions. The code listings for the data interfaces are provided in Appendix A, and for the statistical procedures in Appendix B. Only the code listings relevant to specific problems and solutions will be discussed in detail in this chapter. However, the remaining code listings are provided for completeness.

Table 5.1: Current implemented R packages

R Package	Description	Discussion	Code Listing
sslpiutils	R/PI API interface	Section 5.1	Section A.1
sslgpipi	CVC specific interface to PI	Section 5.1	Section A.1
ssldtutils	Date and time utilities	Section 5.2	Section A.2
ssldbutils	Database interface utilities	Section 5.4	Section A.3
ssldcsutils	Extracting DCS data from local database	Section 5.5	Section A.4
sslxlutils	Generic interface to on-line Excel files	Section 5.5.4	Section A.5
sslgpicvc	CVC specific R code	Chapters 3 and 4	
sslscs	SCS specific R code	Chapter 2	
mltv	Multivariate statistics	Section 5.6.2	Section B.1

All the R packages are documented using the `roxygen2` (Wickham *et al.*, 2014) package by using the `document` function in the `devtools` (Wickham and Chang, 2015) package. Each R function in Appendix A and B is therefore preceded by the relevant markup for the documentation. The output of the documentation is not included as this would be unnecessary duplication. The advantage of using the `roxygen2` package for R documentation is that the documentation and the function are stored in one file, and when the function is updated it is easy to update the documentation. In addition, the documentation is conveniently included in the code listings of all the functions.

5.1 Distributed Control System (DCS) Data Interface

In the petrochemical industry as well as most other process driven industries large volumes of data are generated by various online instruments and analysers. These data are captured and stored in a Distributed Control System (DCS). At Sasol the predominant DCS system for the production facilities is the Honeywell PI system. At the Sasol Group Technology R&T pilot plant facilities the Aspen Process Explorer software is implemented. In this section the interface to the PI system will be discussed, as this study is focused on the process monitoring of the production facilities in Secunda.

The PI software provides a Microsoft Excel Add-In (PI DataLink), a custom interface (PI ProcessBook) as well as a C API interface to the underlying software. In the engineering fraternity the most common interface to the PI system is via the Excel Add-In, as most of the add hoc modelling and monitoring are performed on custom Excel sheets. Although Excel is an adequate solution for short term trending and rapid prototyping, it is not scalable to longer term monitoring and multivariate methods. For real time data downloading Excel do provide access to the raw data (non aggregated), but the inefficiency of downloads, as well as the instability of Excel as a real time in-

terface to the data necessitated the investigation of the PI C API as a faster and more stable interface to the data.

The Industrial Statistics Group (ISG) use the R software (R Core Team, 2015) almost exclusively for statistical analysis and modelling. The R software is written in C and on the Microsoft Windows platform the MINGW compiler (GNU, 2012) is used to compile the R software. Interfacing R to a C dll is therefore well documented and supported. It was therefore decided that R will be used as interface to the PI system via a C dll. This provided for ease of access to the PI data for the statisticians. In addition, R is a scripting language with the facility to run in batch mode. Setting up the software to download data from the PI system on a fixed schedule is easily accomplished via the Microsoft Windows Task Scheduler.

Each measurement in the DCS system is identified by what is known as a "Tag". The process of downloading data therefore starts with identifying the relevant list of tags for the measurements of interest. The data for each tag are stored in a format known as compressed data. This refers to the methodology of capturing and storing of the data.

In essence there are two methodologies:

- The first methodology captures the data at fixed time intervals i.e., a measurement is made every hour, and stored in the DCS system.
- The alternative is capturing the data whenever a change larger than 'n threshold (delta) value takes place on the measured value.

These methodologies are set-points on the DCS system that are decided by the engineers. Each tag contains various attributes, but aside from the description and units the relevant attributes are:

1. Time Stamp - The time of the data point.
2. Value - The value at the time of capturing.
3. Status - The status of the captured value. There are various different statuses, but in practice it can be condensed to either a good value or a bad value. The status of the value is very important for the data aggregation algorithms which will be discussed in Section 5.5.1.

The R package `sslpiutils` was developed to import the data from the PI DCS system into R. The main data download function from the user's perspective is `imppidata` (Listing A.3). The input parameters for the `imppidata` function are a vector of tag values and a begin and end time in character format. The `imppidata` function returns a list of values for the tags. The data aggregation functions require a value before the begin time and after the end time to accurately perform the interpolation for the aggregation algorithm. The data download functions therefore have the functionality to return these values as

well. The format of the list is a list of matrices for each tag with names equal to the tag names. The Value and Status columns of each matrix consist of the values as returned by the PI system. However, the Time Stamp column is converted to seconds from 1-January-1970 similar to the POSIX standard. The advantages of this format will be discussed in Section 5.2. The actual access to the PI API is performed by the C function `getpidata` in Listing A.1. This function encapsulates all the low level detail of the PI API/C interface including memory and error handling. The C function `imppidata` in Listing A.2 encapsulates the conversion between R and C data structures.

The R function `impcurpival` and the matching C functions `getcurpival` and `impcurpival` in Listings A.4 to A.6 are similar to the previous functions except that they only download the most recent data point in the DCS system for the specific tag. These functions are mostly used to check if the data in the local database are up to date.

After downloading the data from the DCS system the data aggregation, statistical analysis and modelling can be performed. The DCS data interface package will differ for each DCS system, and is the only section of the software infrastructure that is system dependent. After the data are imported to R, the format of the data is however identical for any DCS system, and the remaining tool chain can perform without any knowledge of the underlying DCS system. This abstraction of the interface has allowed for the seamless conversion to Aspen Process Explorer and the Honeywell PHD systems for other projects undertaken by the IS group.

The population of the local MySQL database from the PI DCS system happens in two steps. In the first step an empty table is created in the MySQL database and populated from a preset historical time. The second step performs the updating of the local database with any new data from the DCS system. Both these stages are configured via a table called `TagStatus` in the local database. The SQL `CREATE` statement for the `TagStatus` table is provided in Listing 5.1. The columns have the following interpretations:

- `TAGNAME` - The name of the DCS system. The table created in the local database will be named `TAGNAME`.
- `CREATED` - A flag to indicate if the local table has been created.
- `POPULATED` - A flag to indicate if the local table has been populated with the historical data.
- `SDATE` - The start date for the initial population of the local table.
- `UPDATESTEPS` - Due to memory constraints and the high resolution of some of the tags it is not feasible to download all the historical data in one step. This variable indicates the step size in days for the download of historical data.

- `UPDATE` - A flag to indicate if the specific tag should be updated. This is used to turn off the update of a tag if the tag was removed from the DCS system, or if the tag is temporarily not operational.
- `UPDATEGROUP` - The large number of tags in combination with the volume of data necessitates the use of multiple cores to populate and update the local database. This column is used to manually group the variables for update processes.

Listing 5.1: Table TagStatus `CREATE` statement

```
1  CREATE TABLE `TagStatus` ( "TAGNAME" varchar(512) NOT NULL,
2  "CREATED" int(11) DEFAULT NULL, "POPULATED" int(11) DEFAULT NULL
       ,
3  "SDATE" varchar(512) DEFAULT NULL, "UPDATESTEPS" int(11) DEFAULT
4  NULL, "UPDATE" int(11) DEFAULT NULL, "UPDATEGROUP" int(11)
       DEFAULT
5  NULL, PRIMARY KEY ("TAGNAME") )
```

The `buildtags` function in Listing A.7 is responsible for the initial creation of the local table and populating the table with the historical data. In lines 19 to 21 the data from the `TagStatus` table are selected and filtered to include only the rows where the table is either not populated yet, or not created or populated. In lines 24 to 33 the tables that do not exist are created using string substitution into the SQL template in Listing A.9. All these functions are designed to run unattended. Therefore, it is necessary to test the success of each step, and log an error if it occurs. On line 27 the `tblexist` function from the `ssldbutils` package (see Section 5.4) is used to check if the table was successfully created. If the table creation was not successful an error is logged via the `errlog` function in package `ssldbutils`. The `errlog` function adds the details of the error to a specified table in the database.

If the table creation was successful the `CREATED` column is set to 1 for the tag. On line 37 the `inittag` function (Listing A.8) is called to populate the table. On line 27 the current (most recent) value for the tag are imported from PI. The time stamp associated with this value is used as the end time for the `imppidata` function. It is possible to call the `inittag` function with either a `NULL` step size, or an integer value for the number of days at a time to import.

The `imppidata` is used to import the PI data between `bdate` (from the `SDATE` column in the `TagStatus` table) and `edate` (from the call to `impcurpival` on line 27). The error message from PI is checked on line 35 and if it is equal to "Success" the data is exported to the table in the local database using the `exptodbc` function in the `ssldbutils` package. If the `imppidata` returns an error message from PI it is logged using the `errlog` function. Finally the function returns to the `buildtags` function. On line 39 of the `buildtags` function the `tblrows` function is used to check if any data were exported to the local tag table. The `POPULATED` column is set to 1 to indicate that the tag table is

populated. This concludes the initial population step for the tag. This step is only performed once for each tag.

The next step in populating the local tag tables is the updating of the data with new data from PI. Updating the local tag data is performed by the `updgpidb` function (Listing A.10) in the `sslgpipi` package. The function has one input value, `updgroup`. As discussed previously, the large amount of data that needs to be captured necessitates the use of multiple cores in order to complete the action in an acceptable time. It is possible to distribute the tags equally between the number of cores utilised for the updates, but there is large variation in the update frequency of the different tags. Distributing the load equally between the cores is therefore performed manually. The `updgpipi` is called every half an hour using a scheduled Windows batch (`.bat`) file. For each tag the function retrieves the most recent value from PI (line 32) and, using the `getmaxtime` function from the `ssldbutils` package, the most recent value in the local tag table. If there is newer values in PI the data are imported into R and exported to the local tag table. Finally, when all the tags are updated the current time is exported to a table in the database. This value is used to display the time of the most recent update on the MSPEM™ web interface.

5.2 Date/Time utility functions

A major design decision that had to be made early on in the project was the format of date and time storage. After initial research on different standards, it was found that different databases and different DCS systems all use different standards for the date and time storage. As one of the design philosophies is the independence on any specific database, or even DCS system it was decided to use the POSIX standard as guideline, and store the date and time as the number of seconds from 1-January-1970. To allow for the storage of milliseconds, which was a requirement for a project related to filtration, the value is stored as a `double` variable.

The package `ssldtutils` (Section A.2) contains all the date conversion and utility functions. A brief summary of the functions follows:

1. `dateconvrt` (Listing A.11) - Converts a numeric date/time value to a string date/time value.

2. `datestrip` (Listing A.12) - Converts a string date/time value to a numeric date/time value.

3. `exceldateconvrt` (Listing A.13) - Converts a numeric date/time value from Excel to a string date/time value. The Excel date/time format consists of number of days from 1 January 1900. However, it is important to note that Excel erroneously count 1900 as a leap year. This error

originated in the Lotus spreadsheet program, and Excel duplicated the error to be compatible with Lotus.

4. `posixdatestrip` (Listing A.14) - Converts a date/time value in R's internal date/time format to a numeric date/time value.

5. `getcurrtime` (Listing A.15) - Returns the current system time as a string date/time value.

6. `comparedates` (Listing A.16) - Compare two string date/time values, date1 and date2, and returns 1 if date1 is larger than date2, 0 if date1 is equal to date2, and -1 if date1 is smaller than date2.

7. `daysbetween` (Listing A.17) - Return the number of days (rounded) between two dates.

8. `nextday` (Listing A.18) - Advance the time of the input date by 24 hours. This function is normally used to step through days in a loop.

9. `prevday` (Listing A.19) - Decrease the time of the input date by 24 hours.

10. `roundday` (Listing A.20) - Round the input date/time down to the nearest day smaller than the input date/time.

11. `roundhour` (Listing A.21) - Round the input date/time down to the nearest hour smaller than the current hour of the input date/time.

12. `roundnmin` (Listing A.22) - Round the input date/time down to the nearest n minutes smaller than the current value of the input date/time. For example, if n = 15, the date/time value will be rounded down to the nearest quarter of an hour. Therefore, input date/time 1-Apr-15 12:48 will be rounded to 1-Apr-15 12:45.

5.3 Data Storage

Although the data are stored on the DCS system, the speed of access to the system necessitates local storage on the server as the monitoring software, or at the very least in the same geographical area as the server. The IS group is based in Sasolburg and Sasol's major production facilities are based in Secunda which is about 200km from Sasolburg. For reasons that will be discussed in Section 5.7.1 the web server is running the Linux operating system. There is no PI interface on Linux, and therefore the PI interface is located on a separate computer. It was decided to use the MySQL database software for local storage of the data. MySQL is very scalable, and is available on both Windows and Linux. It is also part of the very popular LAMP (Linux Apache MySQL PhP) stack that is responsible for a large percentage of all commercial web sites.

Therefore, it is easy to integrate MySQL into a web based application, and the integration is well supported.

ODBC (Object Database Connection) was used to connect R via the RODBC (Ripley and from 1999 to Oct 2002 Michael Lapsley, 2012) package with MySQL. A major benefit of using ODBC is that, if in future the database is moved to an alternative system, only the connection string in R needs to change, and there will be no impact on the existing software. All the interface functions in R are based on standard SQL code, and are therefore independent of the back-end database. The data are stored as a table for each tag. Each table's name is the tag name. The table consists of three columns, DateTime for the time-stamp, Value for the value, and Status for the status of the tag. The `CREATE` statement for each table is provided in Listing 5.2.

Listing 5.2: Table TagName `CREATE` statement

```
CREATE TABLE `TagName` (
  "DateTime" double NOT NULL,
  "Value" double DEFAULT NULL,
  "Status" varchar(45) DEFAULT NULL,
  PRIMARY KEY ("DateTime")
)
```

Storing the raw data for each tag in the database yields some advantages and disadvantages. A major advantage is the flexibility it provides. Any aggregation statistic over any time period can be performed on the data, and it is therefore possible to change aggregation periods and statistics dynamically. In addition, no data is lost, and all the data are available for future analysis and modelling purposes. This is especially important as some of the production facilities keep data for a fixed period only, and any data older than this period are deleted. The only major disadvantage of storing the raw data is hard drive space. As hard drive space is cheap, it turned out to be no major disadvantage.

There is however a time penalty in creating the aggregated data in real time. After profiling the data aggregation functions it was however determined that these functions are generally not the major bottlenecks in the product. In cases where large amounts of historic data were needed (i.e. for reference sets) the aggregated data were however also stored in the database to alleviate these time constraints. In addition, to further alleviate the aggregation and calculation time penalty the static calculations and graphs are pregenerated as far as possible (see Section 5.6).

The MySQL database currently resides on the same computer as the web server, and this provides for fast data access.

5.4 Generic Database Interface Functions

The package `ssldbutils` (Section A.2) contains most of the functions for the generic database functionality. The following functions are provided:

- `dbconn` (Listing A.23) - Connect to the database and returns an `RODBC` connection object. If any new database type is introduced this function is the only function that needs to be updated.

- `exptodb` (Listing A.24) - Exports an R dataframe to a table in the database. The data are either inserted using the SQL `INSERT` statement, or replaced using the SQL `REPLACE` statement. `INSERT` does not overwrite any data in the table if the primary key is identical, whereas `REPLACE` replaces values if the primary keys are identical. Technically `REPLACE` is identical to a `DELETE` statement followed by an `INSERT` statement. `exptodb` assumes the column names of the R dataframe are identical to the column names of the table in the database.

- `exptodbc` (Listings A.25 and A.26) - This function performs a similar functionality as `exptodb` but the actual exporting of the data is performed in C to improve efficiency. These functions were developed after it was realised that a significant bottleneck in populating the local database with tag data was the exporting of the data from R to the database. Function `exptodbc` can currently only handle numeric data.

- `errlog` (Listing A.27) - As discussed in Section 5.1, error handling is an important aspect of any automated system. The `errlog` function logs an error either to a table in the database when the database is available, or a local comma delimited (csv) file when the connection attempt to the database is not successful. The error message, calling function and the application are logged to the error table. Storing errors in a database is convenient as these functions run on a server, and SQL queries can be used to get aggregated error results on a web page, or locally on the developer's computer. Automatic email notifications are currently being implemented to ensure the developer is notified of errors as they occur.

- `getdbtables` (Listing A.28) - Given a vector of table names and a begin and end time this function returns a list of dataframes with the data from each table between the two times.

- `droptables` (Listing A.29) - Given a vector of table names, it drops the tables from the database schema.

- `tblexist` (Listing A.30) - Returns a 1 if a table exists in the database, and 0 if not.

- `tblrows` (Listing A.31) - Returns the number of rows in a table.

- `getallmax` (Listing A.32) - Given a vector of table names this function returns the maximum date/time in each of these tables.

- `getmaxmax` (Listing A.33) - Given a vector of table names this function returns the overall maximum date/time in the tables.
- `getminmax` (Listing A.34) - Given a vector of table names this function returns the overall minimum of all the maximum date/times in the tables.
- `getmaxtime` (Listing A.35) - Returns the maximum date/time in a table.
- `getmintime` (Listing A.36) - Returns the minimum date/time in a table.

5.5 Local DCS data interface functions

The local DCS data interface functions are provided by the `ssldcsutils` package. In contrast with the packages discussed in the previous sections the `ssldcsutils` package is under constant development. This is largely due to the fact that this package is the interface layer between the underlying data and the developer. The goal of this package is to abstract away the underlying data, and assist the user to focus on the statistical analysis as it relates to the production facility and not the data storage.

5.5.1 Data Aggregation

The format of the raw data discussed in Section 5.3 is very convenient for storage and retrieval, but for statistical analysis the user rarely works with the raw data. This is especially valid for multivariate statistics where, by definition, several variables must be analysed simultaneously. The raw data for the variables are generally not stored at the same time stamps, or even the same frequency. There are several methodologies available for data aggregation. In this section only the methodologies applied in the MSPEM™ application will be discussed. The user interfaces to the DCS systems, for example PI DataLink provide for either the download of raw data, or various time weighted statistics. The engineers use Excel with PI DataLink daily, and any new technology must, at least on a data level, be directly comparable to the results from the PI system. The PI system, and other DCS platforms use the same algorithms to calculate the time weighted statistics with only minor variations on for example the handling of bad data. It is therefore possible to create a set of functions that is independent of the specific DCS system, and can be applied to any raw data. This method of calculating time weighted statistics from stored raw data is called the *aggregate integral* method. It should be noted that this method is only valid for continuous data, and discrete data will be discussed later in this section.

The basis for the integral method is to calculate the statistics from the area under the trend.

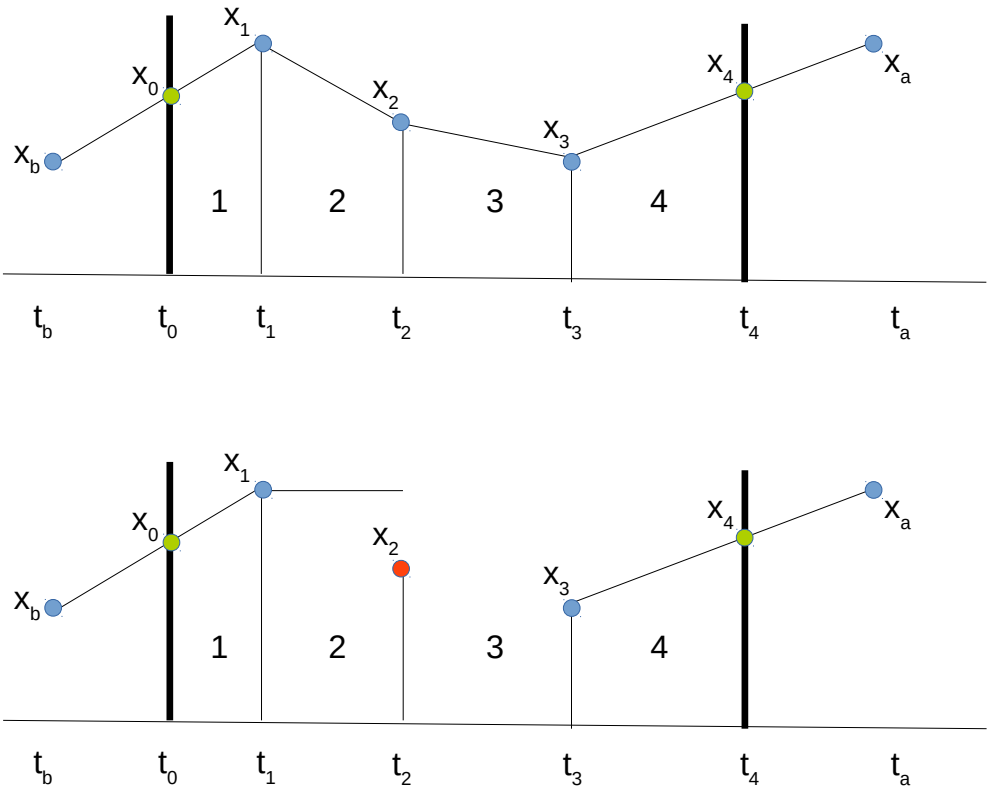


Figure 5.3: Example for time weighted aggregation (red dot indicate bad value, and green dot indicate interpolated value).

The mathematical formula employed for the area under the trend (based on the trapezoidal rule) for trapezoid i is

$$\frac{(x_{i-1} + x_i) * (t_i - t_{i-1})}{2} \tag{5.5.1}$$

where x_{i-1} and x_i are the heights of the two parallel sides (the two consecutive values), and $t_i - t_{i-1}$ is the distance between the values (the time elapsed between the measurements). In addition to the values at each time, there exists a status attribute as well. For simplicity the statuses can be grouped into good and bad. For the PI system a zero status value indicates a good value and any nonzero status indicates a bad value. Figure 5.3 depicts an example of an aggregation time step $(t_0, \cdots, t_4)$ containing three data points inside the aggregation bound (x_1, x_2, x_3), and two values outside the bounds (x_b, x_a). The values x_0 and x_4 are obtained by linear interpolation. For example, x_0 is obtained by linearly interpolating between points x_b and x_1:

$$x_0 = x_b + \frac{x_1 - x_b}{t_1 - t_b} * (t_0 - t_b). \tag{5.5.2}$$

The top graph in Figure 5.3 depicts an example where all the status values are good. The aggregate period therefore contains four trapezoids. In the bottom graph the status value at time t_2 is bad. There are two alternatives for calculating the aggregated values:

1. The first option is to assume the x_1 value stayed constant for the period $t_1 - t_2$. The total "good" time will exclude the period $t_2 - t_3$ (see discussion below). The PI system handles bad values this way.

2. The second option is to use only trapezoids 1 and 4 and assume the total period $t_1 - t_3$ has bad status. Technically this would be more correct, as no good information is available from $t_1 - t_2$, and making the assumption x_1 stays constant over this period is therefore invalid.

In order for the aggregated values to be comparable to the PI output values alternative 1 was implemented.

This discussion of the integral aggregate statistics was partly derived from the Aspen help files. However, the equations were rewritten in statistical notation. The different aggregation values are calculated as follows:

- `good` - The sum of the number of seconds in the aggregation interval with good (0) status values.
- `sum` - The sum of the areas of all trapezoids with good status values.
- `average` - $\frac{sum}{good}$.
- `max` - The largest value with a good status in the aggregation interval.
- `min` - The smallest value with a good status in the aggregation interval.
- `var` -

$$\frac{\sum_{i=1}^{n}(t_{(i)} - t_{i-1})\left[\frac{x_{(i)}-x_{i-1}}{3} + (x_{(i)} - x_{i-1}) * x_{i-1} + x_{i-1}^2\right] - \frac{sum^2}{good}}{good} \quad (5.5.3)$$

  where n is defined as the number of trapezoids in the region.

- `sd` - $\sqrt{var}$.
- `fval` - The first good value in the aggregation period. The value `fval` is used for discrete variables (for example a filter open or closed status variable).

The aggregations statistics are very computing intensive, and were therefore implemented in C (Listing A.38) with an R wrapper function `twstats` (Listing A.39). The statistics are encoded as follows:

- `average` - 0.

- `var` - 1.
- `sd` - 2.
- `max` - 3.
- `min` - 4.
- `fval` - 10.

5.5.2 Data Interface Structure

Although the tag names are the identifiers for any data value on the DCS systems, the format of the values is not optimal from a human interface perspective. An example of a tag name is `B0T4212.PV`. These names are not optimal as it is not possible to infer the actual attributes of the tags from the names. In addition, it is possible for values (especially calculated values) to have more than one tag referring to the same value. Tag names are also sometimes changed as new measurement equipment is installed. It was therefore imperative to design an interface that link the appropriate attributes to a tag. In addition, calculations are often defined based on tag values. A data structure that is recursive i.e., a defined data value can be used as part of the calculation of another defined data value is important, as any change to a tag name etc. is local to the specific data value. Therefore, all the calculations dependent on the value must automatically use the new tag values.

The current solution is based on a data structure encapsulating all the process combinations in a production process similar to Figure 3.1. Specifically, each data value consists of four identifiers:

- `DataDescriptor` - A descriptive name for the variable. This name must be unique to the specific unit, but not necessarily over all the production processes. Some variables are by definition present on all similar production processes, and identical names should then be used. The `DataDescriptor` is compulsory.
- `Side` - The side of the factory (East or West). If a variable is measured outside of the factory or on a variable common to both sides, `Side` is set to 0.
- `Train` - The train the variable is measured on (1,2, 4 or 5 for gasification). If the variable is measured in a process without trains, `Train` is set to 0.
- `Reactor` - The reactor the variable is measured on (GG01 etc. for gasification).

These identifiers are combined into an unique ID with the format:

`##DataDescriptor#Side#Train#Reactor##,`

for example `##O2#West#1#GG01##` for the oxygen measurement on GG01 on Train 1 on the Western factory. The unique ID is automatically created in the Excel input sheet from the inputs. Each unique ID relates to a calculation, which can be any combination of constant values, tags and other data ID's. A 'C_' is added to the front of the tag names in the calculation to facilitate parsing of the calculation. The # character was chosen as separator as it is not possible for an R calculation to contain # values as the # character is the comment character in R. The `CREATE` statement for the `DataStructure` table is provided in Listing 5.3.

The user interface for the population of the data structure is an Excel spreadsheet. The data are exported to the MySQL database via an Excel VBA macro. Excel is a convenient data input interface as most of the calculations are already captured in existing Excel spreadsheets, and most users are comfortable working in Excel. In addition, Excel has very good string handling facilities, and a large portion of the capturing of calculations can be automated.

Listing 5.3: Table DataStructure `CREATE` statement

```
1
2 CREATE TABLE `DataStructure` (
3   `ID` int(11) NOT NULL AUTO_INCREMENT,
4   `DataID` varchar(256) DEFAULT NULL,
5   `DataDescriptor` varchar(256) DEFAULT NULL,
6   `Side` varchar(256) DEFAULT NULL,
7   `Train` varchar(256) DEFAULT NULL,
8   `Reactor` varchar(256) DEFAULT NULL,
9   `Calculation` varchar(5024) DEFAULT NULL,
10  PRIMARY KEY (`ID`),
11  UNIQUE KEY `ID_UNIQUE` (`ID`)
12 );
```

On a high level, two functions are used to extract DCS data from the database i.e., `getdata` and `parsecalcs`. The `getdata` function (Listing A.40) is the function a user will use directly to extract data. Apart from the database names and types the `getdata` function expects the following input parameters:

- `datadesc` - Correlates to the `DataDescriptor` column in the `DataStructure` table.
- `bdate` and `edate` - The start and end data/time for the data download.
- `side` - The side of the factory, `NULL` for all sides.
- `train` - The train, `NULL` for all trains.
- `reactor` - The reactor, `NULL` for all reactors.

- `ssize` - Step size for data aggregation in minutes i.e., 60 for hourly aggregates.
- `stat` - Aggregation statistic (See Section 5.5.1).

For example, downloading the hourly average Oxygen values for all reactors on Train 2 on the Eastern factory between 31 March 2015 and 28 April 2015 the call to the function will be:

```
o2dat <- getdata("O2", "31-Mar-15 00:00:00", "28-Apr-15 00:00:00",
                 calctable, dbnamedat, dbnameapp, dbtype, datetime,
                 side="East", train=2, reactor=NULL, ssize=60, stat=0).
```

The database attributes are all stored in the R package data for each application. One interface improvement for future versions will be to encapsulate all the application specific data in a S3 object that gets instantiated at package load time. This will allow for operating specific variables to be assigned appropriately.

On line 32 of Listing A.40 the `retrexpcal` function (Listing A.41) is called to retrieve the calculation from the database (using the `retrcalcs` function in Listing A.42) and expanding all the sub calculations. On line 32 the `parsetags` function (Listing A.44) is called to extract all the DCS tags from the calculation string. This function uses regular expressions via the R `strsplit` function to split the string on all the operators. All the constants and spaces are then filtered, and a vector of unique tags is returned. The `getdata` function then calls the `retrdcsdata` function on line 42 to return a dataframe with the date/times in the first column, and the aggregated tag data in the remaining columns. The R `with` function is used on line 47 in combination with the `eval` and `parse` functions to calculate the actual results that will be returned. The results are returned as a dataframe with the date/times in the first column, and the returned results in the remaining columns. The column names are derived from the `DataDescriptor` by removing the `##` at the beginning and end, and replacing the `#`'s between the process descriptors with `._.`. For example, the column name for the oxygen variable for GG01 West will be `O2._.West._.1._.GG01`.

The `parsecalcs` function (Listing A.50) is used to retrieve the most recent data for a calculation. This function is mostly used for populating dials and flow-sheets which requires a single value for each data point.

5.5.3 Caching

Although a web based interface is very convenient as a medium to present results to users it does add certain challenges to the development of a product. A major challenge is the expectation of response speed. In a product like Microsoft Excel users are accustomed to a certain degree of lag when a

calculation is requested via the push of a button or the changing of a value in a cell. However, most users expect a website to react almost instantaneously to a request. Although the `getdata` function is computationally efficient (most of the underlying processing, for example the data aggregation, is performed in C), the data retrieval and aggregation can lead to a noticeable delay on the real-time web application. For example, the calculation of the GPI discussed in Chapter 4 necessitates the request and aggregation of 15 minute average data over a 24 hour period for 11 variables over 84 gasifiers. Two approaches are possible to alleviate this delay:

- Calculating the aggregation offline and storing the results in the database. The calculation will therefore obtain the data directly from the aggregation tables.
- Caching the calculated data on the hard drive, and retrieving the cached results on demand.

Both of these approaches are currently implemented for the MSPEM™ product, but for the CVC and SCS applications caching is used almost exclusively. One of the major limitations in the R interface is a lack of support for the `BLOB` (binary large objects) data type. Support for the `BLOB` data type would have enabled the storage of R objects directly in a data table. It is currently possible to store R objects indirectly in the database by serialising (to character data) and conversely un-serialising the character data (to the original R object). However, due to size limitations in the storage and transfer of character data this solution was also not feasible. It was therefore decided to focus on caching the data, as this method was easier to implement, more efficient, and more general. One disadvantage of file system caching is the lack of transactional support build into SQL databases. It is therefore theoretically possible that the files can become corrupted and the data will be lost (although highly unlikely on modern file systems). However, as the only data that are cached are derived data (calculated from the raw data stored in the database) the risk (and cost) of the corruption of cached data is negligible. Memoization is a form of caching that occurs on the functional level i.e., a function will cache the results of a calculation given specific input values, and if the same input values occur again it will return the same results (without recalculation).

Various alternatives were investigated for the implementation of data caching in R. Ultimately, an approach based on the `SOAR` package (Venables and based on original code by David Brahm, 2013) was selected due to the following advantages:

- The `SOAR` package is based on the lazy loading facility in R, and the cached objects are therefore not loaded into memory when not needed. Given the size of the objects, and the fact that R is memory based, this is a big advantage.

- The cached objects are attached to the R search path, which simplifies the interface to the data.
- Arguably the biggest advantage of the `SOAR` package is that the code is well written and well maintained. It was therefore possible to easily understand the underlying implementation to optimise the interfaces. In the unlikely event of the package becoming deprecated it will be possible to maintain a local copy of the package.

On a high level the `SOAR` package consists of two functions relevant to this study:

- `Attach` - Attaches a cache folder to the R search path. The default position on the search path is 2. Function `Attach` also create lazy loaded objects for each of the files in the cache folder. Therefore, the objects are available on the search path but will only be loaded when needed.
- `Store` - The `Store` function saves an object to the cache folder, adds it to the relevant search path, and removes the object from the global environment. The file name of the object in the cache folder is the object name with one exception to cater for different case sensitivity rules on different file systems (Windows is not case sensitive .i.e., FILENAME is identical to filename, whereas Linux, and all other Unix based operating systems including Mac OS X, are case sensitive). `SOAR` changes capital letters to lowercase letters with an `@` symbol preceding the letter. For example, FileName would be stored as `@file@name`.

The data caching was implemented via a wrapper function `getcacheddata` (Listing A.47). The interface to the `getcacheddata` function is identical to the `getdata` function except for two extra parameters:

- `cachpath` - The path to the applications cache folders.
- `cachname` - The specific folder in the `cachpath` folder to use. Therefore, it is possible to have more than one cache per application.

The `getcacheddata` function returns identical results to the `getdata` function, and can therefore be used as a drop-in replacement.

Designing a caching strategy is not a simple endeavour. There are various trade-offs to consider. Some of the design decisions and choices were:

1. The granularity of the cached objects. Specifically, using the gasifiers as an example for this issue, should the variables for each gasifier be cached separately, or should the variable as such be stored for all gasifiers. For example, if oxygen is cached, should separate objects be stored for GG01 and GG02, or per train etc. A higher granularity will potentially be more

efficient if only data for one gasifier are retrieved. Retrieving data for subsequent gasifiers will however all be separate retrievals. However, if the oxygen data for all gasifiers are stored in a single object, the retrieval of oxygen for all gasifiers only will be slightly less efficient (unnoticeable in real live application) than the retrieval for a single gasifier. In this study the decision was made to store the data for a single variable, for a single aggregation step and type in one object. For example, the oxygen data for a 15 minute time weighted average aggregation for all gasifiers will be stored in an R dataframe consisting of 85 columns. The first column will be the data/times, and the subsequent columns will be the aggregated oxygen data for each gasifier.

2. The cache object naming convention. The naming convention for the cached objects is important both to ensure the uniqueness of each identifier, and as a method to filter out objects cached via the `getcacheddata` (or similar) interface, and general cached objects. Specifically, this is important for cache maintenance (which will be discussed later in this section). For example, the XRF stack simulation models discussed in Section 2.3.3 are also computing intensive and the results of the simulation models are cached as objects containing all the relevant input and output with the Heap ID as name. These cached objects should not be confused by the cache maintenance routine with the cached DCS data. The convention used for the cached DCS data is as follows:

   ```
   DataDescriptor._.AggregationStep._.AggregationType.
   ```

   The 15 minute time weighted average oxygen data will therefore be stored in an object named `O2._.15._.0`.

As R is based on the Scheme programming language (Ihaka and Gentleman, 1996), it is possible to do computations on the language objects themselves. This is called meta programming or non-standard evaluation (Wickham, 2014*a*). This powerful feature makes the caching framework implemented in `SOAR` possible. However, it does sometimes lead to unintended complexities when interfacing with the software in a non-standard way. Similar to the call to R's library function, the `SOAR` package allows for the calling of the functions without quoting of the path names. This leads to complexities if the path is inside a variable name. For example, the `Attach` function will interpret a call `Attach(cachpath,cachname)` as literally meaning the path "cachpath", and the cachname "cachname". It was therefore necessary to make use of the same non-standard evaluation programming methodology used by `SOAR` (and for example the R `library` function) to attach the appropriate cache (See line 36 in Listing A.47). To code and encode the different names some utility functions (`.cnamedec` and `.cnameenc`) were written (Listing A.48).

After attaching the cache to the search path the `getcacheddata` function first encode the data descriptor, step size and statistic (line 37), and then test if the cached object exists (line 38). If it exists, it retrieves the actual object into memory (line 39). It then checks if the cached object's maximum date is smaller than the requested dates, or the minimum date is larger than the requested dates (lines 42 and 53 respectively). If either one of these conditions is true, the missing data are retrieved via the `getdata` function, appended to the cached data, and stored in the cache (lines 45 - 49 and 56-60 respectively). Note that the call to the `getdata` function sets the side, train and reactor to `NULL` indicating that the data for all the reactors on both sides should be returned. The cached object is then filtered for the requested dates and side, train, reactor combination (line 69) and returned after detaching the cache from the search path (lines 88 - 90). If the cached object does not exist the data will be retrieved via `getdata` (line 72) and stored in the cache (line 75).

The implementation of the caching methodology had an order of magnitude impact on the efficiency for data retrievals. An additional benefit is that the cached data are very convenient for offline development. One potential problem with cached data is that the DCS system is not always functioning as intended. It is possible (and does happen) that data for a specific tag are not updated or available for a certain period, and then the value stored in the cache will be static. If the tag data become available the cached data will not reflect the data calculated in real-time. The `cachemaintenance` function (Listing A.49) was implemented to periodically delete a specified range of historical data from the cache and recalculate and cache the data. The `cachemaintenance` function filters the objects in the cache and only retains objects containing the `._.` sequence of characters in the object name using the `iscalc` function in Listing A.48. The names are decoded using the `.cnamedec` function. The remaining code functions are similar to the code in `getcacheddata` except that a specified set of data is first deleted from the object before the object is populated again until the current time.

5.5.4 Excel Interface Functions

As discussed in Chapter 2 various Excel files containing data are manually updated by plant operators daily. These data must be captured and integrated with the DCS data for effective plant monitoring. The `sslxlutils` package was created to automatically capture the data from Excel files residing in various files in various locations on the intranet servers.

The `sslxlutils` package consists of three functions.

- `trawlxls4data` (Listing A.54) - This is the main interface function that parses the inputs, and opens the Excel workbooks for reading. Initially, the `XLConnect` package was used to connect to the Excel files, but due to memory constraints in the Java libraries used by `XLConnnect` to interface

to Excel a conversion to the `openxlsx` package was performed. The `openxlsx` package uses `Rcpp` to connect via C++ to the Excel API.

- `xlstrawlworker` (Listing A.55) - This function reads the data from the open Excel workbooks and ensures that the data is in the appropriate format for the following function.

- `createtagdatafromxls` (Listing A.56) - This function takes the data from `xlstrawlworker` and converts it to a data structure identical to the tag (DCS) data discussed in Section 5.5

Therefore, these functions open the Excel workbooks on the remote servers, capture the data, convert it to a format similar to the tag data, and store the data in the database. The resulting table in the database can be used exactly like the DCS data, and all the functions used to retrieve DCS data i.e., the functions in the `ssldcsutils` can be used to retrieve the captured and converted Excel data.

The user interface to the `sslxlutils` package is an Excel spreadsheet. The inputs from the Excel spreadsheet are exported to the MySQL database table `exceltrawlertbls`. The `CREATE` statement for the `exceltrawlertbls` is provided in Listing 5.4. Both the date and the data must be extracted from the Excel files. In addition, in some of the cases both the folder and the file names are related to the current data. For example, the Excel file for 28 April could be stored in the following format `./Apr-15/28-Apr-15.xls`. In some of the Excel files more than one data value per day is captured, and a time-stamp for each value must be imported as well. This information is captured in the database (and therefore the Excel interface) as follows:

- `Tag` - The name of the table in MySQL where this data must be stored.

- `Area` - The area in the plant this data belongs to i.e., CVC or SCS in this study.

- `FileIdentifier` - A unique file identifier created by the user. This identifier is used to ensure that a file is opened only once, and all the data is extracted while it is still open. Therefore, there can be more than one row with the same file identifier in the table.

- `FileName` - The name of the Excel file. This name can include R date conversion characters. For example, one of the files captured is (where the actual path to the folder on the server is omitted):

  ```
  .\Production log Sheet %Y\%m-%b\CTF Log - %d %b %Y.xlsx
  ```

  This file is therefore in a folder ending with the current year, sub folder with the month in both numeric and three letter format. The name of

the file contains the numeric day of the month, the month in numeric format, and the year with the century. The `trawxls4data` function gets the latest date in the database, and then loops through the days from that day (or some predetermined days earlier to ensure any changes that were applied to the files subsequently are captured) to the current day, and substitutes the time stamp values into this format string to determine the actual file name.

- `RawFileName` - Flag variable to indicate if the file name contains date and time conversion characters.
- `SheetName` - The sheet name in Excel where the relevant data reside.
- `StartRowData` and `EndRowData` - The rows in the sheet where the relevant data reside.
- `StartColData` and `EndColData` - The columns in the sheet where the relevant data reside.
- `DateType` - The type of the date. Currently three values are valid:
  - `CURRENT` - The date is the same as the date used to get the file name.
  - `FIXED` - The date is fixed. This should never happen.
  - `READ` - The date is read from the Excel file.
- `TimeType` The type of the time. Currently four values are valid:
  - `CURRENT` - The time is the same as the time used to get the file name.
  - `FIXED` - The time is fixed. For example "00:00".
  - `READ` - The time is read from the Excel file.
  - `DATE` - The time is part of the date field read from the Excel file.
- `DateValue` - The value for the date if `DateType` is `FIXED`.
- `TimeValue` - The value for the time if `DateType` is `FIXED`.
- `DateFormat` - The format of the date if the date is read from the Excel sheet.
- `TimeFormat` - The format of the time if the time is read from the Excel sheet.
- `StartRowDate` and `EndRowDate` and `EndRowData` - The rows in the sheet where the relevant dates reside.

- `StartColDate` and `EndColDate` - The columns in the sheet where the relevant dates reside.
- `StartRowTime` and `EndRowTime` and `EndRowData` - The rows in the sheet where the relevant times reside.
- `StartColTime` and `EndColTime` - The columns in the sheet where the relevant times reside.
- `NumDaysOffset` - The number of days to retrieve again to make sure that any later changes in the Excel files are captured.

Listing 5.4: Table exceltrawlertbls `CREATE` statement

```
1
2  CREATE TABLE `exceltrawlertbls` (
3    `Tag` varchar(256) NOT NULL,
4    `Area` varchar(256) NOT NULL,
5    `FileIdentifier` varchar(256) NOT NULL,
6    `FileName` varchar(2156) NOT NULL,
7    `RawFileName` smallint(2) DEFAULT '1',
8    `SheetName` varchar(256) NOT NULL,
9    `StartRowData` int(11) DEFAULT '0',
10   `EndRowData` int(11) DEFAULT '0',
11   `StartColData` int(11) DEFAULT '0',
12   `EndColData` int(11) DEFAULT '0',
13   `DateType` varchar(256) NOT NULL,
14   `TimeType` varchar(256) NOT NULL,
15   `DateValue` varchar(256) DEFAULT NULL,
16   `TimeValue` varchar(256) DEFAULT NULL,
17   `DateFormat` varchar(256) DEFAULT NULL,
18   `TimeFormat` varchar(256) DEFAULT NULL,
19   `StartRowDate` int(11) DEFAULT '0',
20   `EndRowDate` int(11) DEFAULT '0',
21   `StartColDate` int(11) DEFAULT '0',
22   `EndColDate` int(11) DEFAULT '0',
23   `StartRowTime` int(11) DEFAULT '0',
24   `EndRowTime` int(11) DEFAULT '0',
25   `StartColTime` int(11) DEFAULT '0',
26   `EndColTime` int(11) DEFAULT '0',
27   `NumDaysOffset` int(11) DEFAULT '0',
28   PRIMARY KEY (`Tag`)
29 );
```

5.6 Statistical Programming

The essence of the MSPEM™ methodology is to extract, visually represent and provide insight from the large volumes of data captured in the production facilities. The field of multivariate statistics and data visualisation provides a

rich set of tools to achieve this goal. It was decided early in the project that `R` (R Core Team, 2015) is well suited to the statistical programming intent of this project as it provides a rich set of statistical methodologies, flexibility for the implementation of new methodologies, and powerful graphical capabilities. The `R` package system is another major benefit, as it provides a convenient vehicle for modularisation and re-usability.

5.6.1 Multivariate Graphics

There exist various packages to create biplots, but currently the two most complete packages are:

- `BiplotGUI` (la Grange *et al.*, 2009) - This package provides the user with a powerful graphical user interface, and a high level of graphical interactivity.
- `UBbipl` (`https://dl.dropboxusercontent.com/u/17860902/UBbipl_3.0.4.tar.gz`) (Gower *et al.*, 2011) - This package is by far the most advanced, complete and powerful biplot package currently available in R (and arguably in any software). In contrast to the `BiplotGUI` package the `UBbipl` package provides a command line interface to the functions. This is a major advantage for a real time on-line application, as well as general exploratory statistical analysis. In addition, the package is well documented in the accompanying book "Understanding Biplots" (Gower *et al.* (2011)).

However, web based applications pose some additional challenges:

- The graphs are generated without any human intervention. It is not possible to adjust axes, or any other display attributes before displaying the graph.
- Users expect a certain level of interactivity from a web based graphic. A web based graphic should therefore at the minimum provide for the implementation of:
  - Mouse hover events (tooltips) - Hovering with the mouse pointer over a graphical object provides some information. For example, date value of a point, mean value of a group etc.
  - Mouse click events (drill down) - Clicking on a graphical object displays more information about the underlying data. For example, clicking on a point in a monitoring biplot displays a bar chart with the contributions to performance deviation.

- Web based statistical graphics pose an additional challenge in the trade-of between the size of the graphic and the resolution. Generally, statistical software provides facilities for vector graphics in the form of pdf and ps/eps files, and various forms of raster graphics. Raster graphics do not scale well, and can lead to prohibitively big files at higher resolution.

The SVG graphical format (`http://www.w3.org/Graphics/SVG/`) has emerged as the preferred format for web based graphics (Eisenberg and Bellamy-Royds, 2014). Some of the advantages are:

- SVG is an XML based format developed by the World Wide Web Consortium specifically for web based graphics. Because SVG is XML based the information is stored in plain text that can be edited in any text editor. In addition, it is easy to generate and edit SVG files programmatically using any XML capable language.
- SVG graphics is fully integrated in the browsers DOM (Document Object Model). Therefore, it is possible to interact with the graphics via JavaScript events (including mouse events). In addition, the graphic can be updated in real time via JavaScript i.e., a new point can be added to a biplot without redrawing the plot.
- SVG is a vector graphic, and is therefore fully scalable i.e., zooming into a graphic will retain the resolution of the graphic.
- SVG graphics can be animated. For example, it is therefore possible to animate the addition of points to a biplot to give an indication of the movement of a process over time.

R is capable of generating SVG graphics via the `gridSVG` package (Murrell and Potter, 2015). The `gridSVG` package was developed as an extension to the `grid` package (Potter, 2013). The grid graphics system is a powerful graphics system which is used in both of R's main graphic extension packages, `ggplot` and `lattice`. The grid graphic system provides an object based graphic system that allows for the interrogation of each graphic object to obtain dimensions. In addition, each graph can be provided with hooks which allow for the re-computation of sizes when the parent object size change. This allows for the programming of fully dynamic graphical objects. The grid graphics system is discussed in detail in Murrell (2011).

An alternative approach for web based statistical graphics is to do the calculations in R, and use a JavaScript graphing library, for example D3 (Bostock *et al.*, 2011; Murray, 2013). In addition, it is possible to combine a JavaScript library with the SVG output from `gridSVG`. Three approaches to the generation of the interactive graphics are possible:

1. A purely R based approach, where the SVG graphic is generated via `gridSVG`. The `gridSVG` package allows for the embedding of the JavaScript

events via the `grid.garnish` function. The actual JavaScript code can be embedded in the SVG via the `grid.script` function. The `grid.animate` function can be used to add animation to the graphic.

2. A purely JavaScript based approach using a library like D3.

3. A combined approach where the initial SVG is generated in R, but additional interactive elements are added via the JavaScript D3 library.

All three these approaches expect the developer to be proficient in at least R and JavaScript. In addition, for option 2 and 3 a high level of proficiency with a JavaScript graphing library is a prerequisite. A disadvantage of option 2 is the loss of the graphing capability for offline R data analysis. It was therefore decided to start with the development of a purely R based approach (using JavaScript events for interactivity) with the option of future migration to option 3. One important consideration for the generation of SVG graphics (and grid graphics in general) is to follow a naming convention (Murrell, 2012; Potter, 2013). The JavaScript events are linked to object names in the SVG. Any external JavaScript library will also interact with the SVG object via the ID's of the graphical objects. The `gridSVG` package uses the names of the grid objects to create the SVG ID's for the objects.

Listing 5.5: `gridSVG` example

```
1
2  ##' Plot graph inside heatmap
3  ##'
4  ##' Utility function to plot a graph inside a heatmap layout.
5  ##'
6  ##' @param data Data to plot
7  ##' @param fill Fill colour
8  ##' @param name Name for the plotting viewport
9  ##' @param yscale Y scale for the viewport
10 ##' @param yat Y tick marks
11 ##' @param layout.pos.row layout.pos.row if added to viewport with
       layout
12 ##' @param layout.pos.col layout.pos.col if added to viewport with
       layout
13 ##'
14 ##' @export
15
16 plotminigraph <-
17 function(data, fill, name=NULL, yscale=c(0,1), yat=NULL, layout.pos.row
     =NULL,
18           layout.pos.col=NULL, event=NULL) {
19   vp1 <- viewport(layout.pos.row=layout.pos.row, layout.pos.col=
       layout.pos.col, name=name)
20   pushViewport(vp1)
21   grid.rect()
22   if(is.null(yat)) {
23     margins = c(0,0,0,0)
24   } else {
25     margins=c(0.5,3.5,0.5,0.5)
26   }
27   vp2 <- plotViewport(margins=margins, name=paste(name, "dvp", sep=""
       ),
28                       xscale=c(0, length(data)+1), yscale=yscale)
29   pushViewport(vp2)
30   grid.rect(x=0,y=0,width=1,height=1,just=c("left","bottom"), gp=
       gpar(fill=fill, col="black"),
31             name=paste(name, "rect", sep=""))
32
33   if(!is.null(yat)){
34     for(i in yat)
35       grid.lines(x=c(0,1), y=unit(c(i,i), "native"), gp=gpar(col="
           lightgrey", lwd=0.25))
36     grid.yaxis(at=yat, gp=gpar(col="black"))
37   }
38
39   grid.lines(x=unit(1:length(data), "native"), y=unit(data, "native")
       )
40   grid.points(x=length(data), y=data[length(data)], pch=19, size=unit
       (0.5, "char"))
41   grid.lines(x=c(0,1), y=unit(c(mean(data, na.rm=T), mean(data, na.rm=
       T)), "native"),
```

```
42                 gp=gpar(col=gray(0.1),lty="dashed"))
43   grid.text(x=0.5,y=0.1,label=name,gp=gpar(cex=0.75),name=paste(
         name,"text",sep=""))
44   if(!is.null(event)) {
45       grid.garnish(paste(name,"rect",sep=""),onmousedown=event,"
             pointer-events"="all",
46                    title=name,cursor="pointer")
47       grid.garnish(paste(name,"text",sep=""),onmousedown=event,"
             pointer-events"="all",
48                    title=name,cursor="pointer")
49   }
50   popViewport(2)
51
52 }
```

An example of using the gridSVG package is provided in Listing 5.5. This function creates the sub graph inside the larger GPI graphic discussed in Chapter 4. Of specific interest is:

- Line 31 and line 44 - The naming of the graphical objects (line 31 for the rectangle object and line 44 for the text object).
- Lines 45 to 48 - The use of the `grid.garnish` function to add:
  - A mouse click event, `onmousedown=event`, where `event` is an R variable containing the name of a JavaScript function to call.
  - A tooltip, `title=name`, where name in this example is the name of a gasifier.
  - A changed mouse pointer `cursor="pointer"`.

Figure 5.4 depicts an example of the mouse hovering over GG27 on the GPI graphic.

5.6.2 Discussion of the `mltv` package

The R code to generate the PCA and CVA biplots discussed in Chapters 3 and 4 is provided in Section B.1. A design goal of the `mltv` package was to separate as much of the calculation into separate functions, and develop a generic plotting function that will be appropriate for any multivariate biplot i.e., PCA and CVA biplots for this study. The `mltv` package is still under active development, and this discussion and code listings will cover a snapshot of the package. The `mltv` package has been influenced by the various existing biplot packages, in particular `UBbipl`.

5.6.2.1 PCA and CVA Algebra

The functions for calculating the PCA and CVA algebra are:

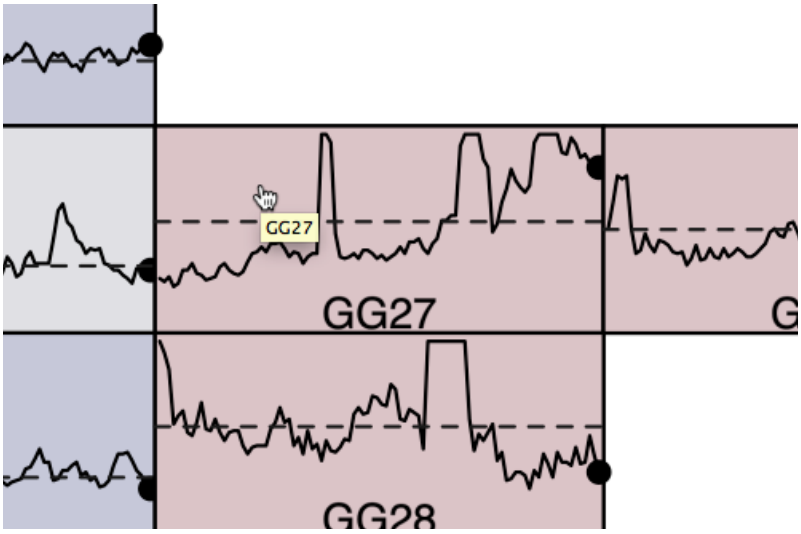


Figure 5.4: GPI example of SVG Graphic

- PCA - The `mltvpca` function (Listing B.2) implements the PCA algebra as discussed in Section 3.3.1. The input parameters are:
  - `X` - Data Matrix.
  - `g` - Optional grouping variable.
  - `evec` - Eigenvectors to include.

  The `mltvpca` function returns a S3 object of class `mltv_pca`, inheriting from the `mltv` class. The S3 class contains the following slots:
  - `data` - Original input data (`X`).
  - `evec` - Eigenvectors to include.
  - `colmeans` - The mean value for each column of `X`.
  - `colsd` - The standard deviation for each column of `X`.
  - `groups` - The groups assigned to each row of `X`.
  - `axes` - Variable axes to include in biplot.
  - `axnames` - Names of variable axes.
  - `vraxes` - Loadings to use for axes marker projections.
  - `vrpoints` - Loadings to use for new sample projections.
  - `projmean` - Projected mean values.
  - `projpoints` - Projected values for samples (rows of `X`).
  - `quality` - PCA quality.
  - `scaled` - Corresponds to the `scalemat` parameter.
  - `eigen` - Eigenvectors corresponding to the `evec` parameter.

  - `sigma` - All the calculated eigenvalues.
  - `loadings` - All the calculated loadings.
  - `datcent` - The centred `X` data.

- CVA - The `mltvcva` function (Listing B.2) implements the CVA algebra as discussed in Section 3.4.3. The input parameters are:
  - `X` - Data Matrix.
  - `g` - Vector of groups.
  - `evec` - Eigenvectors to include.
  - `weightedCVA` - Weights to use for CVA, either "weighted","unweightedI" or "unweightedCent" as discussed in Gower *et al.* (2011).

The `mltvcva` function returns a S3 object of class `mltv_cva`, inheriting from the `mltv` class. The S3 class contains the following slots:

- `data` - Original input data (`X`).
- `evec` - Eigenvectors to include.
- `colmeans` - The mean value for each column of `X`.
- `colsd` - The standard deviation for each column of `X`.
- `groups` - The groups assigned to each row of `X`.
- `axes` - Variable axes to include in the biplot.
- `axnames` - Names of variable axes.
- `vraxes` - Loadings to use for axes marker projections.
- `vrpoints` - Loadings to use for new sample projections.
- `projmean` - Projected mean values.
- `projpoints` - Projected values for samples (rows of `X`).
- `quality` - CVA quality in the original variables.
- `classification` - Classification prediction accuracy.
- `transformmat` - Transformation matrix for original `X` values to the reduced canonical space.
- `transformmatinv` - Transformation matrix form reduced canonical space to original space.
- `weights` - CVA weights for groups.
- `scaled` - Always `FALSE` but included for axis calculations.
- `datcent` - The centred `X` data.

5.6.2.2 Data projection functions

The code for the projection of new samples onto the orthogonal axes consists of one generic S3 method, and the corresponding S3 method for a specific class:

- `project` (Listing B.3) - A generic S3 method to return the projection on the orthogonal axis for a object of class `mltv` with input parameters:
  - `mltv` - Object of one of the `mltv` classes.
  - `xnew` - New data to project.
  - `g` - Optional grouping variable.
- `project.mltv_pca` (Listing B.4) - S3 method that returns the projection of a new sample onto the orthogonal PCA axis with the following input parameters:
  - `mltv_pca` - Object of the `mltv_pca` class.
  - `xnew` - New data to project.
  - `g` - Optional grouping variable.

  The method returns the `mltv_pca` object with a new slot, "xnew" containing the following:
  - `data` - Original input data (`xnew`).
  - `groups` - The groups assigned to each row of `xnew`.
  - `colmeans` - The mean value for each column of `xnew`.
  - `colsd` - The standard deviation for each column of `xnew`.
  - `projpoints` - Projected values for samples (rows of `xnew`).
  - `datcent` - The centred `xnew` data.

5.6.2.3 Measures of fit for axes

The code for the calculation of the measures of fit for the axes consists of three generic S3 methods, and the corresponding S3 methods for specific classes:

- Axis Predictivity:
  - `axispredictivity` (Listing B.5) - generic S3 method for calculation of axis predictivity with the following inputs:
    - `mltv` - Output from one of the class `mltv` functions.
    - `axlim` - Cutoff value for the inclusion of axes in the biplot.
  - `axispredictivity.mltv_pca` (Listing B.6) - S3 method for the `mltv_pca` class to calculate the axis predictivity as discussed in Section 3.3.2 with the following input parameters:

    * `mltv` - Output from the `mltvpca` functions of class `mltv_pca`.
    * `axlim` - Cutoff value for the inclusion of axes in the biplot.

  The output of the function is an object of class `axispred_predictivity` inheriting from class `axispred` containing the following slots:

    * `axispred` - Calculated axis predictivity values.
    * `axispredlimit` - Cutoff point for axis inclusion.

  The `axispred_predictivity` object is inserted in a slot named "axispred" in the `mltv_pca` object, and the `mltv_pca` object is returned.

- `axispredictivity.mltv_cva` (Listing B.7) - S3 method for the `mltv_cva` class to calculate the axis predictivity as discussed in Section 3.4.3.1 with the following input parameters:

    * `mltv` - Output from the `mltvcva` functions of class `mltv_cva`.
    * `axlim` - Cutoff value for the inclusion of axes in the biplot.

  The output of the function is an object of class `axispred_predictivity` inheriting from class `axispred` containing the following slots:

    * `axispred` - Calculated axis predictivity values.
    * `axispredlimit` - Cutoff point for axis inclusion.

  The `axispred_predictivity` object is inserted in a slot named "axispred" in the `mltv_cva` object, and the `mltv_cva` object is returned.

- Axis mspe:

  - `axismspe` (Listing B.8) - generic S3 method for calculation of axis mspe with the following inputs:

    * `mltv` - Output from one of the class `mltv` functions.
    * `axlim` - Cutoff value for the inclusion of axes in the biplot.

  - `axismspe.mltv_pca` (Listing B.9) - S3 method for the `mltv_pca` class to calculate the axis mspe as discussed in Section 3.3.2 with the following input parameters:

    * `mltv` - Output from the `mltvpca` functions of class `mltv_pca`.
    * `axlim` - Cutoff value for the inclusions of axis in the biplot.

    The output of the function is an object of class `axispred_mspe` inheriting from class `axispred` containing the following slots:

    * `axispred` - Calculated mspe values for the axes.
    * `axispredlimit` - Cutoff point for axis inclusion.

    The `axispred_mspe` object is inserted in a slot named "axispred" in the `mltv_pca` object, and the `mltv_pca` object is returned.

  - `axismspe.mltv_cva` (Listing B.10) - S3 method for the `mltv_cva` class to calculate the axis mspe as discussed in Section 3.4.3.1 with the following input parameters:
    * `mltv` - Output from the `mltvcva` functions of class `mltv_cva`.
    * `axlim` - Cutoff value for the inclusion of axes in the biplot.

    The output of the function is an object of class `axispred_mspe` inheriting from class `axispred` containing the following slots:
    * `axispred` - Calculated mspe values for the axes.
    * `axispredlimit` - Cutoff point for axis inclusion.

    The `axispred_mspe` object is inserted in a slot named "axispred" in the `mltv_cva` object, and the `mltv_cva` object is returned.

- Axis inclusion:
  - `axisinclude` (Listing B.11) - Generic S3 method to determine axis inclusion with the following input parameter:
    * `axisinclude` - Object of class `axispred`.
  - `axisinclude.axispred_predictivity` (Listing B.12) - S3 method to determine axis inclusion for objects of the `axispred_predictivity` class with the following input parameter:
    * `axispred` - Object of `axispred_predictivity` class.

    The output of the function is a numeric vector of axes to include in the biplot.
  - `axisinclude.axispred_mspe` (Listing B.13) - S3 method to determine axis inclusion for objects of the `axispred_mspe` class with the following input parameter:
    * `axispred` - Object of `axispred_mspe` class.

    The output of the function is a numeric vector of axis to include in the biplot.

5.6.2.4 Data enclosure functions

Three different data enclosure algorithms are implemented in the `mltv` package:

- Alpha bags:
  - `alphabag` (Listing B.15) - Calculates the x and y coordinates for the alpha bags of a data set given:
    * `x` - x values of the data set.
    * `y` - y values of the data set.
    * `alpha` - alpha value for the bag.

The `alphabag` function returns the x and y values for the alpha bag of the data set. Note that this function calls the `abagplot` Fortran function. This function was obtained from the `UBbipl` package source (`https://dl.dropboxusercontent.com/u/17860902/UBbipl_3.0.4.tar.gz`).

- `createbags` (Listing B.14) - The `createbags` function calls the `alphabag` function for each group in the data. The input parameters are:
  - `mltv` - An object of class `mltv`.
  - `alpha` - alpha value for the `alphabag` function.

  The function returns the `mltv` object with an additional slot "container" containing a list of the unique group names in `mltv$groups`, and each group containing the x and y values for the alpha bags.

- Concentration ellipse:
  - `concellipse` (Listing B.17) - Calculate the x and y coordinates for the concentration ellipses of a data set given:
    - `x` - x values of the data set.
    - `y` - y values of the data set.
    - `alpha` - alpha value for the ellipse.

    The `concellipse` function returns the x and y values for the concentration ellipse of the data set.
  - `createconcellipse` (Listing B.16) - The `createconcellipse` function calls the `concellipse` function for each group in the data. The input parameters are:
    - `mltv` - An object of class `mltv`.
    - `alpha` - alpha value for the `concellipse` function.

    The function returns the `mltv` object with an additional slot "container" containing a list of the unique group names in `mltv$groups`, and each group containing the x and y values for the concentration ellipses.

- Convex hull:
  - `convexhull` (Listing B.19) - Calculates the x and y coordinates for the convex hulls of a data set given:
    - `x` - x values of the data set.
    - `y` - y values of the data set.
    - `layer` - layer value for the convex hull.

    The `convexhull` function returns the x and y values for the convex hull of the data set.

- `createconvexhull` (Listing B.18) - The `createconvexhull` function calls the `convexhull` function for each group in the data. The input parameters are:
  - `mltv` - An object of class `mltv`.
  - `layer` - layer value for the `convexhull` function.

  The function returns the `mltv` object with an additional slot "container" containing a list of the unique group names in `mltv$groups`, and each group containing the x and y values for the convex hulls.

5.6.2.5 Axis label and marker calculations

The code to calculate the axes, markers and labels of the biplot axes proposed by Gower and Hand (1996) is provided by the following two functions:

- `calcgowaxes` (Listing B.20) - Creates an object with all the information needed to draw the biplot axes given:
  - `mltv` - Object of one of the `mltv` classes.

  The function returns the `mltv` with an additional slot "gowaxes", containing:
  - `axcoef` - The axes coefficient values.
  - `axes` - for each of the axes:
    - `slope` - The slope of the axis.
    - `side` - Which side of the plot the axis label should be.
    - `at` - Where on the side of the plot should the axis label be.
    - `markangle` - The angle of the tick marks on the axis.
    - `markerpos` - The position of the tick marks, and labels for the axis.
    - `markers` - The numeric labels for the tick marks.
- `axmarkerseq` (Listing B.21) - Utility function called by `calcgowaxes` to calculate the position of the tick marks and labels of each axis given:
  - `mltv` - Object of one of the `mltv` classes.
  - `ind` - Numeric value to indicate the axis to calculate tick mark positions for.
  - `n.int` - Numeric value to indicate how many tick marks are desired for the specified axis.
  - `scl` - Boolean value to indicate if the data are scaled.

  The `axmarkerseq` function returns a numeric vector of `n.int` values between the upper and lower limits for the `ind`-th biplot axis.

5.6.2.6 Biplot theme functions

The aesthetics of the plots generated by the `mltv` package are governed by a theme. The functions to create the default theme are:

- `createmltvtheme` (Listing B.22) - Creates a sensible default theme given:
  - `mltv` - An object of one of the `mltv` classes.

  The output of the `createmltvtheme` function is an object of class `mltv_theme` containing the following slots:
  - `fill` - The background fill colour of the plot.
  - `titlecol` - The colour of the plot title.
  - `axes`
    - `lables`
      - `padding` - Size of the padding to use around biplot axis labels.
    - `colfunc` - A function that generates colours for the biplot axes given n, the number of axis.
  - `groups`
    - `colfunc` - A function that generates colours for the groups given n, the number of groups.
    - `pch` - - A function that generates pch values for the groups given n, the number of groups.
  - `container`
    - `alpha` - alpha (transparency) value for the containers (data enclosures).
  - `points`
    - `pchfunc` - A function that generates pch values for the points given n, the number of groups.
    - `alpha` - alpha (transparency) value for the points
  - `plot`
    - `axes` - Boolean value to indicate if the biplot axes should be plotted.
    - `container` - Boolean value to indicate if the container (data enclosures) should be plotted.
    - `means` - Boolean value to indicate if the group means should be plotted.
    - `points` - Boolean value to indicate if the biplot points should be plotted.

    * `newpoints` - Boolean value to indicate if the projected points for the new sample data should be plotted.
  - `xpadding` - A factor to increase the x axis range beyond the range of the plotted data.
  - `ypadding` - A factor to increase the y axis range beyond the range of the plotted data.

- `biplotguicolors` (Listing B.23) - Return colours similar to the default colours of the `BiplotGUI` package given:
  - `n` - Number of colours to return.
- `ggplotcolors` (Listing B.24) - Return colours similar to the default colours of the `ggplot2` package given:
  - `n` - Number of colours to return.
- `pchfunc` - Return pch values given:
  - `n` - Number of pch values to return.

5.6.2.7 Biplot utility functions

- `calcextvals` (Listing B.26) - Calculate extreme values for the plot from the data and theme settings given:
  - `mltv` - Object of the `mltv` class.
  - `mltvtheme` - Object of the `mltv_theme` class.

  The `calcextvals` function returns the `mltv` object with an additional slot "extvals", containing a vector of extreme values c(xmin, xmax, ymin, ymax) in `native` units inserted in "extvals" slot of the `mltv` object.
- `gmembermat` (Listing B.27) - Creates a group membership matrix given:
  - `g` - A vector of length n with group membership for each sample.

  The `gmembermat` function returns a $j \times n$ indicator matrix, where j is the number of groups. The matrix contains 1's in the j_i-th column of the matrix for each n_i in group j_i and zeros otherwise.
- `maxstring` (Listing B.28) - Returns the length of the longest string given:
  - `labels` - A vector of string labels.

  The `maxstring` returns a grid unit object returned by the max function.
- `getaxlabels` (Listing B.29) - Changes the biplot axes labels to a more convenient format for the plotting routine given:

  - `mltv` - An object of class `mltv`.

  The `getaxlabels` function returns the `mltv` object with an additional slot "axlables", containing:

  - `side1` - A list with all the axes labels located at side 1 containing the label location in default units (0-1).
  - `side2` - A list with all the axes labels located at side 2 containing the label location in default units (0-1).
  - `side3` - A list with all the axes labels located at side 3 containing the label location in default units (0-1).
  - `side4` - A list with all the axes labels located at side 4 containing the label location in default units (0-1).

5.6.2.8 Biplot plotting functions

The code for the biplot plotting function consists of two S3 methods:

- `mltvbipl` (Listing B.30) - Generic S3 methods for generating a multivariate plot given:
  - `mltv` - An object of class `mltv`.
  - `mltvtheme` - An object of class `mltv_theme`.
  - `main` - A title for the plot.
- `mltvbipl.mltv` (Listing B.31) - An S3 method to generate a multivatiate plot given:
  - `mltv` - An object of class `mltv`.
  - `mltvtheme` - An object of class `mltv_theme`.
  - `main` - A title for the plot.

  The function `mltvbipl.mltv` generates the plot, and return a list containing:

  - `mltv` - The original `mltv` object updated with internal plot calculations.
  - `mltvtheme` - The original `mltvtheme` object.

  The `mltvbipl.mltv` method consists of the following sections:

- Line 20 - The extreme values are calculated by calling the `calcextvals` function.

- Lines 23 to 26 - The `axes` slot is updated to include only the axes which comply to the measure of fit criterion if the `axispred` slot exists in the `mltv` object. In addition, the axis names are updated to include the value for the measures of fit.
- Line 28 - the `calcgowaxes` function is called to retrieve the relevant data for the creation of the axes.
- Lines 30 to 44 - The plotting device and data viewport for plotting are created and populated.
- Lines 46 to 62 - The colours for the axes are retrieved and the axis lines and markers are plotted. The `getaxlabels` function is called to retrieve the data for the plotting of the labels of the axes.
- Lines 65 to 99 - The containers, points, projected points and means are plotted if the appropriate theme plot values are set to `TRUE`.
- Lines 100 to 149 - Viewports are created for each of the four margins of the plot and the relevant labels and the title of the plot is added.

To demonstrate the use of the `mltv` package to create biplots, the creation of a PCA biplot with concentration ellipse with axis predictivity calculations, and a CVA biplot with alpha bags and axis predictivity calculations will be demonstrated.

PCA biplot demonstration Given a $n \times p$ data set $\boldsymbol{X}$, and a $m \times p$ new data set $\boldsymbol{Xnew}$ the following sequence of function calls in Listing 5.6 will generate the biplot in Figure 5.5.

- Line 2 - The PCA analysis is performed.
- Line 4 - The calculations for the concentration ellipse are performed with default alpha value (`alpha=0.90`).
- Line 6 - The new data are projected on the orthogonal PCA axes.
- Line 8 - The theme object is created.
- Line 10 - The axis predictivities are calculated.
- Line 12 - The biplot is generated.

Listing 5.6: PCA biplot demonstration

```
1
2     pcabipl <- mltvpca(X)
3
4     pcabipl <- createconcellipse(pcabipl)
5
6     pcabipl <- project(pcabipl,Xnew)
7
8     pcabipltheme <- createmltvtheme(pcabipl)
9
10    pcabipl <- axispredictivity(pcabipl,axlim=0.35)
11
12    mltvbiplout <- mltvbipl(pcabipl,pcabipltheme)
```

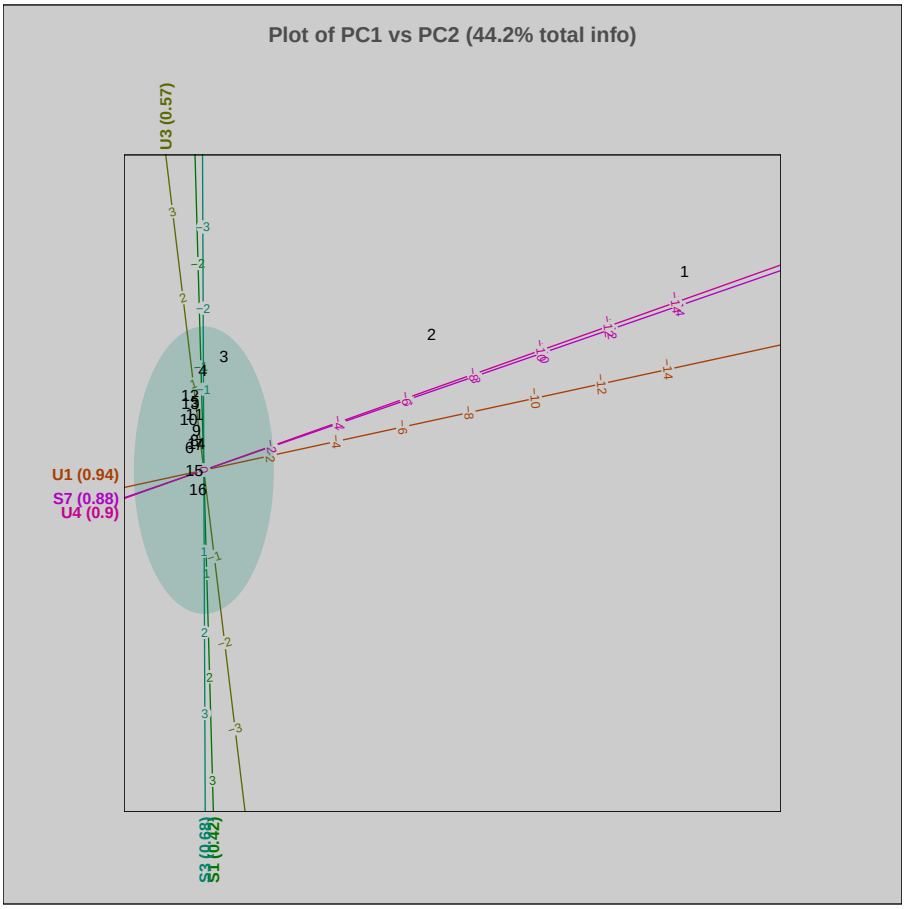


Figure 5.5: PCA Biplot demonstration

CVA biplot demonstration Given a $n \times p$ data set $\boldsymbol{X}$, and a grouping vector $\boldsymbol{g}$ of length n the following sequence of function calls in Listing 5.7 will generate the biplot in Figure 5.6.

- Line 2 - The CVA analysis is performed.
- Line 4 - The calculations for the alpha bags are performed with default alpha value (`alpha=0.90`).
- Line 6 - The axis predictivities are calculated.
- Line 8 - The theme object is created.
- Line 10 - The biplot is generated.

Listing 5.7: CVA biplot demonstration

```
1
2   cvabipl <- mltvcva(X,g)
3
4   cvabipl <- createbags(cvabipl)
5
6   cvabipl <- axispredictivity(cvabipl,axlim=0.5)
7
8   cvabipltheme <- createmltvtheme(cvabipl)
9
10  cvabiplout <- mltvbipl(cvabipl,cvabipltheme)
```

5.6.2.9 GOPA analysis

The code for the GOPA analysis discussed in Section 3.2.3 is provided in Listing B.32. This code accompanied the article Coetzer *et al.* (2014). The Listing consists of three functions:

- `GOPA` - The functions that performs the GOPA optimisation given:
  - `Xk` - List of matrices for GOPA analysis.
  - `K` - Number of matrices in Xk.
  - `pk` - Vector of length K consisting of ncol for each matrix in Xk.
  - `istoropic` - Should isotropic scaling be performed.
  - `Pk.scaling` - Should pk-scaling be performed.
  - `eps` - Tolerance parameter for GOPA algorithm
- `DrawBiplotAxes` - A utility function to draw the biplot axes.
- `GOPAplot` - Creates the PCA biplot of the GOPA output given:
  - `GOPA.out` - The output from the GOPA function.
  - `axes` - Should biplot axes be added to the plot.
  - `legpos` - Where should the legend be positioned. This parameter is sent directly to the `legend` function in R, and should therefore be of the same format.

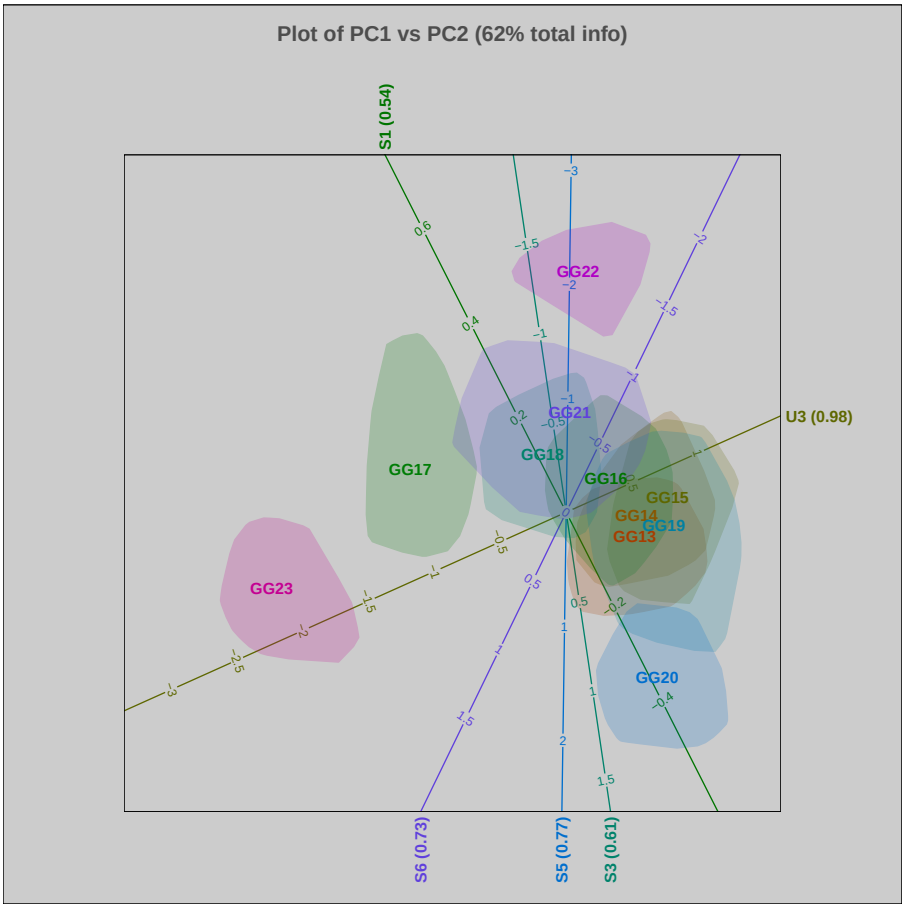


Figure 5.6: CVA Biplot demonstration

5.7 User Interface

Various strategies for user interfaces can be employed. A custom executable module can be developed, spreadsheet software like Microsoft Excel can be utilized, or a web interface can be developed. A web interface yields various advantages, some of which are:

- *Ease of use* - Most users are familiar with a web interface, and will therefore intuitively know how to navigate a web based application.
- *Maintainability* - Any software that resides on the client's computer needs to be updated when bugs are fixed, or new versions become available. A web interface ensures that all changes are immediately available to all relevant users as the application resides on a central server. In addition, in any modern information technology landscape, installing custom software on users' computers is not allowed. A web based interface ensures that no software will be installed on the users computers.

- *Interactive* - Web technologies like JavaScript, JSON and AJAX make it possible to have highly interactive web based applications.
- *Smart Devices* - Smart devices like smart phones and tablets are becoming part of the information technology landscape. A web based interface makes the application available on these smart devices without any additional software development. In addition, the core application will be identical for use on any device as long as the development is standard compliant.

5.7.1 Dynamic Web Programming

Various technologies are available for the development of interactive web applications. For this study PHP on an Apache web server was chosen as the preferred server side language. PHP is widely available, and well supported on all major operating systems. In addition, the Rserve R library (Urbanek, 2012) supports PHP as one of the languages that can interface with R. The link between the MySQL database and PHP is also well supported and part of the default installation of PHP.

Listing 5.8 provides an example of PHP code generating a dynamic page of dials for a dashboard, given an single input.

- Line 3 - A variable sent via the website link is captured via the PHP `$_GET` command and stored in the `DashPage` variable. Note that PHP variable names are preceded by a $ character.
- Lines 6 to 13 - A SQL query is dynamically created using PHP string concatenation and the returned result (if avaialable) is used to set the `title` variable.
- Lines 15 to 22 - Anothter MySQL query is executed to return the most recent time stored in the database, and the header of the page is created (lines 21 and 22).
- Line 24 - The connection to R is established via Rserve and stored in a variable.
- Line 28 - The `sslssfdb` library is loaded.
- Line 29 - The content of the `DashPage` variable is sent to the R function `creategraphs` and `creategraphs` is executed. The `creategraphs` function creates dials for the dashboard given the `DashPage` and the most recent data in the database.
- Lines 31 to 43 - The MySQL table "DialPlots" is queried to find the appropriate dials images to load into the page. In this example the html `map` attribute is used to add a link to a dial. This was necessary because

Internet Explorer 8 was the standard browser, but it is not compatible with SVG graphs.

Listing 5.8: PhP R and MySQL Interface Example

```
1
2  <?php
3  $DashPage = $_GET['DashPage'];
4  include_once 'ssfdbfunctions.php';
5
6  $query = "SELECT `MenuTitle` FROM `MenuTable` WHERE `MenuID`=" . $
       DashPage . ";";
7  $qres = queryMysql($query);
8  if (mysql_num_rows($qres) > 0){
9    $qtitle = mysql_fetch_array($qres,MYSQLI_ASSOC);
10   $title = $qtitle['MenuTitle'];
11 } else {
12   $title = $DashPage;
13 }
14
15 $query = "SELECT MAX(`DateTime`) FROM `lastupd`;";
16 $qres = queryMysql($query);
17
18 $qltime = mysql_fetch_row($qres);
19 $lasttime = $qltime[0];
20
21 echo "<H1 ALIGN=LEFT>Sasol Synfuels Dashboard - " . $title . " ("
       . Date("d-M-Y H:i", $lasttime) . ")</H1>";
22 echo "<H3 ALIGN LEFT>Updates Every 15 Minutes </H3>";
23
24 $s = Rserve_connect();
25 if ($s == FALSE) {
26   echo "Connect FAILED";
27 } else {
28   Rserve_eval($s, "library(sslssfdb)");
29   $fsum = Rserve_eval($s, "creategraphs(" . $DashPage . ")" );
30
31   $query = "SELECT `ID`, `Link` FROM `DialPlots` WHERE `DashName`=
         " . $DashPage . ";";
32   $qres = queryMysql($query);
33   for($i = 0; $i < mysql_num_rows($qres);$i++){
34     $title =  mysql_fetch_row($qres);
35     if (file_exists($wd . "/" . $title[0] . ".jpeg")){
36       echo "<a><img src='" . $wd . "/" . $title[0] . ".jpeg'
             height=280 width=280 usemap='#" . $title[0] . "map'></a>"
             ;
37       echo "<map name='" . $title[0] ."map'>
38       <area shape='poly'
39       coords='140,140,95,220,110,230,140,235,170,230,185,220'
40       href=http://" . $webhost .
41       "/SSFDashBoard/SSFTrend.php?DashPage=" . $DashPage . "&ID='
             " . $title[0] . "'&Type=DialPlots alt='Trend'>
```

```
42          <area shape='rect' coords='0,0,280,280' href=" . $title[1] .
                ">
43          </map>";
44
45      }
46
47  }
48 }
49 ?>
```

One disadvantage of creating the graphs and tables on demand, is that a noticeable time delay can occur when the user open the page, or click on a different menu item. An alternative approach to create real time or close to real time web based interfaces is to pre-generate all the graphs and tables at a fixed interval i.e. every half an hour. There are some advantages and disadvantages to this approach.

- Advantages:
  - The user experience no delay on the page when loading or on clicking a menu item.
  - Each graph and table is generated only once for all users. This can have a large impact on server load.
  - More complex graphics can be generated. Creating complex SVG graphics can be very computationally expensive. Pre-generating these graphics are therefore a necessity.
- Disadvantages:
  - All the graphs and tables on the website are generated every half an hour. This can lead to a large number of graphs being created periodically. For example, generating the trend graphs for the gasifiers for eleven variables, $11 \times 84 = 924$ graphs are generated every half an hour (excluding all the other graphs on the website).

The current implementation of the website in this study is implemented as pre-generated graphics wherever possible. Applications involving user input can obviously not be pre-generated, and must therefore be created on demand. Referring to Figure 5.2, the websites are currently hosted on an IBM xServer 3650 M4 server with 24 cores and 64Gb of ram. Centos 6.3 64 bit is installed on the server. Therefore, a possible solution to the time consuming generation of graphics is to make use of parallel processing. The type of problems encountered in this study due to the 84 production processes generally fall under the parallel processing category known as “embarrassingly parallel”. Embarrassingly parallel problems generally consist of many similar independent computations i.e., the generation of 924 trend graphs for gasification. R has various packages available that offer parallel processing solutions. Two

of the most popular packages for parallel processing are `snow` and `parallel`. An advantage of the `snow` package is that it works on all operating systems. The major advantage of the `parallel` package is that it is part of the base R package since R2.14.0. It makes use of process forking to generate parallel processes, which have a very low overhead and is therefore very efficient. Forking is not available on the Microsoft Windows platform, and the parallel package is therefore limited to one core on Windows. However, this is not a major disadvantage as the production and development servers utilised in this study are all running Centos Linux. The code to pre-generate the graphics therefore make use of the parallel package, and is scheduled to run every half an hour using the Linux `Cron` scheduling program.

In the next sections a brief overview will be provided of the web based interfaces to the Sasol Coal Supply application discussed in Chapter 2 and the Coal Gasification application discussed in Chapters 3 and 4.

5.7.2 Sasol Coal Supply Interface

Figure C.1 provides a screen grab of the MSPEM™ Sasol Coal Supply menu structure. An overview of all the pages and user interface structures will now be provided. Note that due to confidentiality constraints only menu items discussed in Chapter 2 will be discussed here.

- *Home* (Section C.1.1) - The home page provides an overview of the XRF output combined with the material movement files (see Section 2.1.3) for the last 28 days by default. The range of XRF data to display can however be changed by selecting new dates on the drop down calendars depicted in Figure C.2 and clicking on the "Update Information" button. The home page consists of six tabs:
  - *Ash Overview* - The ash overview (depicted in Figure C.3) provides a longer term overview of the ash per mine. A histogram, table of ash properties, and trend graph are provided. This page is generated on demand, as the user can change the begin and end date via the calendar inputs.
  - *Ash Per Mine* - The ash per mine tab (Figure C.4) depicts a histogram of the individual mines over the selected period.
  - *Material Movement* - The material movement tab (Figure C.5) displays a bar chart of the percentage of each mine that was sent to each stacker yard over the selected time period.
  - *Ash Elements (Oxides)* - The ash elements tab displays additional ash elements captured by the XRF.
  - *Elemental Sulphur* - The elemental sulphur tab displays the calculated elemental sulphur results from the XRF output.

  - *Organic Matter* - The organic matter tab displays the (inferred) organic matter content over the selected time period.

- *Heap Information* (Section C.1.2) - The heap information menu item provides information about the general information for each heap for all the stackers. A specific heap (or multiple heaps) is selected via the selection box shown in Figure C.6. When the "Upload" button is clicked the information for the heap is displayed as shown in Figure C.7 for Heap 15374. The creation of the heap information was discussed in Section 2.3.3 and the output shown in Figure C.7 is identical to the information in Tables 2.4 and 2.5.

- *Heap Reclaim Simulation* (Section C.1.3) - The interface to the heap reclaim simulation is identical to the interface to the heap information (Figure C.8). On pressing the "Run Simulation" button the simulation discussed in Section 2.3.3 is performed for the selected heaps, and the output shown in Figure C.9 provided. The output for Heap 15374 is identical to Figures 2.20 and 2.22.

- *Reclaimer Simulation* (Section C.1.4) - The reclaimer simulation page provides the user with the opportunity to simulate the predicted ash properties of the coal going to gasification given a specific reclaiming strategy. Figure C.10 depicts the selection box to select the heaps that will be reclaimed in the simulation. On pressing the "Load Simulation" button an input screen is presented for each heap giving the user the opportunity to select the current reclaimer position on the heap, the reclaiming direction, and the tons/hour that will be reclaimed for the heap. Figure C.11 shows the input screen for Heap 15374. On pressing the "Run Simulation" button the graph for the heap is updated to indicate the actual portion reclaimed (Figure C.12) and a graph of the predicted ash going to gasification given the inputs is provided (Figure C.13). This simulation in combination with the stacker simulation provides the user with the tools to plan a reclaiming strategy to optimise the reclaimed coal quality for gasification stability.

- *Heap Reports* (Section C.1.5) - The heap reports menu provides the user with the facility to select multiple Heaps, and generates a pdf report containing the information in both the Heap Information and the Heap Reclaim Simulation pages. The report is generated via the R `knitr` package (Xie, 2013, 2014, 2015). This report is distributed daily by the CVC coordinator to the relevant people.

5.7.3 Coal Gasification Interface

Figure C.15 provides a screen grab of the MSPEM™ Sasol Gasification and CVC menu structure. An overview of the pages relevant to this study and

user interface structures will now be provided. Note that due to confidentiality constraints only menu items discussed in Chapters 3 and 4 will be discussed in detail. In addition, in the screen grabs discussed below the graphs on the actual data are substituted with graphs on the centred and scaled data.

- *Home* - The home page contains a table with the current values (last 15 minutes) for the most important gasification operating parameters. In addition, line graphs for the last 24 hours are provided for some of these parameters.
- *GPI* (Section C.2.2) - The GPI page contains the Gasifier Performance Index for the Western and Eastern Factory. The page for each side consists of four tabs (Figure C.16):
  - *GPI* - This tab contains the GPI graph (Figure C.17) as discussed in Section 4.3.2. As demonstrated in Section 5.6.1 the user can click on each cell of the graph to populate the remaining tabs with the information for the specific gasifier.
  - *GPI Contribution* - This tab displays the contribution plot for the last 15 minutes for the specified gasifier. Figure C.18 depicts the contribution plot for GG17.
  - *GPI Time Series* - This tab contains the time series plots for the selected gasifier over the past 24 hours (Figure C.19).
  - *GPI Multivariate Plots* - This tab contains the monitoring biplots (Figure C.20) as discussed in Section 4.3.4.
- *CQI* - The CQI page contains a Coal Quality Index and the input values to give an overall indication of the quality of the coal sent to gasification. Note, the CQI was not discussed in the current study due to IP restrictions on the parameters used in the CQI.
- *CVC Reports* - The CVC reports menu contains various coal related reports and graphs in both plain html, and optional pdf (via `knitr` (Xie, 2013, 2014, 2015)) format. The pdf reports are distributed daily by the CVC coordinator to the relevant people.
- *GG Reports* - The GG reports page contains the facility to generate a gasification report for either the Easter or Western Factory for a specified date. This report is generated via `knitr` (Xie, 2013, 2014, 2015) and contains detailed information about the performance of gasification.
- *Plant Overview* - The Plant Overview page provides an overview on the specific performance parameters for each side of the factory.

- *Longer Term Monitoring* (Section C.2.2) - The Longer Term Monitoring page provides access to the CVA biplots (Figure C.21) as discussed in Section 4.3.5. Currently only the first two principal components are depicted for each train.

5.8 Conclusions

In this chapter an overview of the software infrastructure for the MSPEM™ application was provided. This infrastructure was developed according to the criteria of:

- Flexibility.
- Scalability.
- Fit for purpose.
- Independence.

A high level overview of the infrastructure as implemented is provided in Figure 5.1.

In Section 5.1 the interface functions to the PI DCS from R were discussed. These functions are part of the `sslpiutils` R package. The C API interface functions as well as the R wrapper functions were presented. In addition, the R functions and MySQL tables for the management of the local copy of the DCS data were presented. These functions allow for the automated creation of local copies of the tag data, the population of these tables, and periodic updates of these tag tables from the DCS system.

In Section 5.2 the date/time utility functions in the `ssldtutils` package were discussed. These functions encapsulate the conversion of various date/-time formats. In addition, some utility calculation functions are provided. The base date/time format these functions operate on is the POSIX standard i.e., number of seconds from epoch (1-Jan-1970).

In Sections 5.3 and 5.4 the structure of the local storage MySQL database was presented, as well as the generic database interface R package `ssldbutils`. The `ssldbutils` package provides wrapper functions for the most common database interface commands i.e., connecting to the database, importing and exporting data, dropping tables, and querying for table existence and the number of rows in the tables. In addition, functions are provided to find various combinations of the minimum and maximum dates in the local DCS storage tables.

In Section 5.5 the `ssldcsutils` R package was presented as an interface to the local DCS data. An important functionality provided by the `ssldcsutils` package is the capability to perform various data aggregation functions. The topic of data aggregation as applicable to DCS data was discussed in Section

5.5.1. In Section 5.5.2 a data interface structure was presented that abstracts away the direct interface to the data via the name of the tags. This structure provides for a human readable naming convention to access the data. In addition, this structure is recursive, and the human readable names can be used in calculations to define new interface structures. Finally, in Section 5.5.3 a caching strategy is presented. This strategy strives to increase the efficiency of data access.

In Section 5.5.4 the `sslxlsutils` R package was discussed. This package allows for the extraction of Excel data from various locations on the Sasol Intranet, and the storage of the data in the local MySQL database in a format identical to the local DCS data. These data can therefore be used identically to the DCS data.

In Section 5.6 the topic of statistical programming was discussed, and more specifically, the topic of multivariate graphics. Various existing packages were presented. The advantages of a `grid` based approach to generate SVG graphics via the `gridSVG` R package were discussed. Specifically, SVG graphics allow for interactive web graphics with various mouse events via JavaScript. An R, JavaScript, and combined approach for the generation of interactive graphics were discussed and contrasted. Finally, in Section 5.6.2 the `mltv` R package was presented. The `mltv` package consists of various functions to build up a multivariate graph in steps. The plotting functions are isolated from the underlying multivariate algebra, and can therefore be applied to any multivariate statistic that provides output in appropriate format. This package is under constant development, and will in future be made available via CRAN or github.

Finally, in Section 5.7 the user interface to the MSPEM™ applications was presented. The advantages and disadvantages of a web based interface were presented, as well as the use of PHP as dynamic web programming language. The interface between PHP, R and MySQL was demonstrated via an example function. In Sections 5.7.2 and 5.7.3 the websites for the Sasol Coal Supply and Sasol Gasification and CVC MSPEM™ applications were presented via various screen grabs from the websites.

In conclusion, this chapter presented the software infrastructure implementation and design decisions for the implementation of the work discussed in this study. Underlying the statistical results presented in this study is the need for real time efficient access to various data sources, and the integration of these data. This aspect of statistics provides an enormous challenge for the statistician if not designed and implemented in an efficient and standardised manner. Only after the efficient access to integrated and standardised data has been resolved can the statistician apply his skills to the extraction of insight from the data. Significant software development efforts were required to implement the complete real-time multivariate monitoring application. In addition, the presentation of results to the user is often partly to be blamed for the lack of acceptance of statistical methodology in industry. It is of the utmost impor-

tance to strive for a clear, intuitive and visually appealing presentation of the results to foster an adoption of statistical methodology in industry. In this chapter, the current solution to these challenges has been presented. However, although these results serve as a foundation for future work, continued research and development are required to fully resolve these challenges.

Chapter 6

Conclusions and Future Research

The developed real-time multivariate process monitoring MSPEM™ application has been implemented for the entire coal value chain from Sasol Coal Supply up to and including Coal Gasification. The development and implementation of the MSPEM™ real-time multivariate process monitoring application addressed the following challenges:

- The efficient integration and standardisation of diverse data sources.
- The development and implementation of appropriate multivariate statistical methodologies for real-time multivariate data analysis and process monitoring.
- The development of an efficient web based interface.

6.1 Conclusions

In Chapter 2, the focus was on the integration of various data sources with statistical and Data Science techniques in order to generate real-time coal quality information. It was demonstrated how the on-line measurements from one XRF analyser situated on stacker four on the Eastern side can be used in combination with Excel files from SGS Laboratories, Material Movement Data from SCS, Stockpile Information files and reclaimer tonnage information from the PI DCS system to generate real time information on the coal quality of the heaps on the stack yard.

A stacker simulation model was developed and implemented to predict the properties of the heaps on all the stack yards utilising the XRF data in combination with the other data sources. This stacker simulation model allows for the prediction of coal properties (including but not exclusively to ash) over the length of the heap, as well as the average ash percentage and standard error of the ash percentage for the heap. The information is used to do blend planning for the week, and is compared to the laboratory analysis data for validation purposes.

The output from the stacker simulation model was combined with the data from the material movement files, and the reclaimer data from the PI DCS system to calculate the reclaimed ash and standard deviation of the ash sent to the gasification plant. The information provides for the integration between the Sasol Coal Supply real-time information and the Sasol Gasification facility. The reclaimer simulation model was implemented in a module which is utilised to specify the coal heaps for a reclaiming strategy for minimising the impact of sub-optimal coal qualities on the gasification factory. The integrated data at the SCS facilities have therefore been converted into valuable insight which is used for making informed decisions.

Finally, the application of a design and analysis of computer experiments strategy has been applied to an existing FactSage (Bale *et al.*, 2009) computer model for predicting the slagging property of the ash. The input to the slagging model consists of ash elemental properties that are obtained from the XRF analyser. Due to the compositional nature of the ash elemental data, a space filling design appropriate for mixture experiments was required. Various dissimilarity measures appropriate to mixture experiments were compared using two design comparison criteria. The Divergence dissimilarity measure was found to be optimal as a dissimilarity measure for creating space-filling designs for mixture experiments. In addition, an optimisation strategy was presented for obtaining maximin designs. The optimal dissimilarity measure and optimisation strategy were applied to specify the experimental design. The model runs were obtained for the design points, and a quadratic model was fitted to the data. The predicted results from the quadratic model compared very well to the actual results, and it was therefore concluded that the design was appropriate for the model. The developed model is used to carry out real-time prediction of the slagging property of the ash.

In Chapter 3, the focus was on the development and implementation of a real-time multivariate statistical process evaluation and monitoring methodology for the Sasol Coal Gasification facility. The Sasol Coal Gasification plant is a highly complex facility consisting of two separate facilities known as Gasification West and Gasification East. Each facility consists of four trains, each containing between 10 and 11 gasifiers. Each gasifier is equipped with instrumentation which records on-line performance data on the gasifiers.

The first step in a multivariate statistical evaluation and monitoring process is the selection of an appropriate reference data set. A novel application of Generalised Orthogonal Procrustes Analysis (GOPA) was proposed and demonstrated as a criterion for the selection of the optimal train and the optimal combination of the number of weeks as the reference set for all the production processes. PCA analyses and biplot displays were used to visualise and to interpret the results from the GOPA analysis. These interpretations provide important insight into and quantification of the relationships between the variables on the production facility. The application of GOPA for reference set selection, and the accompanying PCA analyses are new in the multivariate

process monitoring literature. The steps proposed for determining the optimal reference set for multivariate monitoring of multiple production processes are summarised in Figure 6.1.

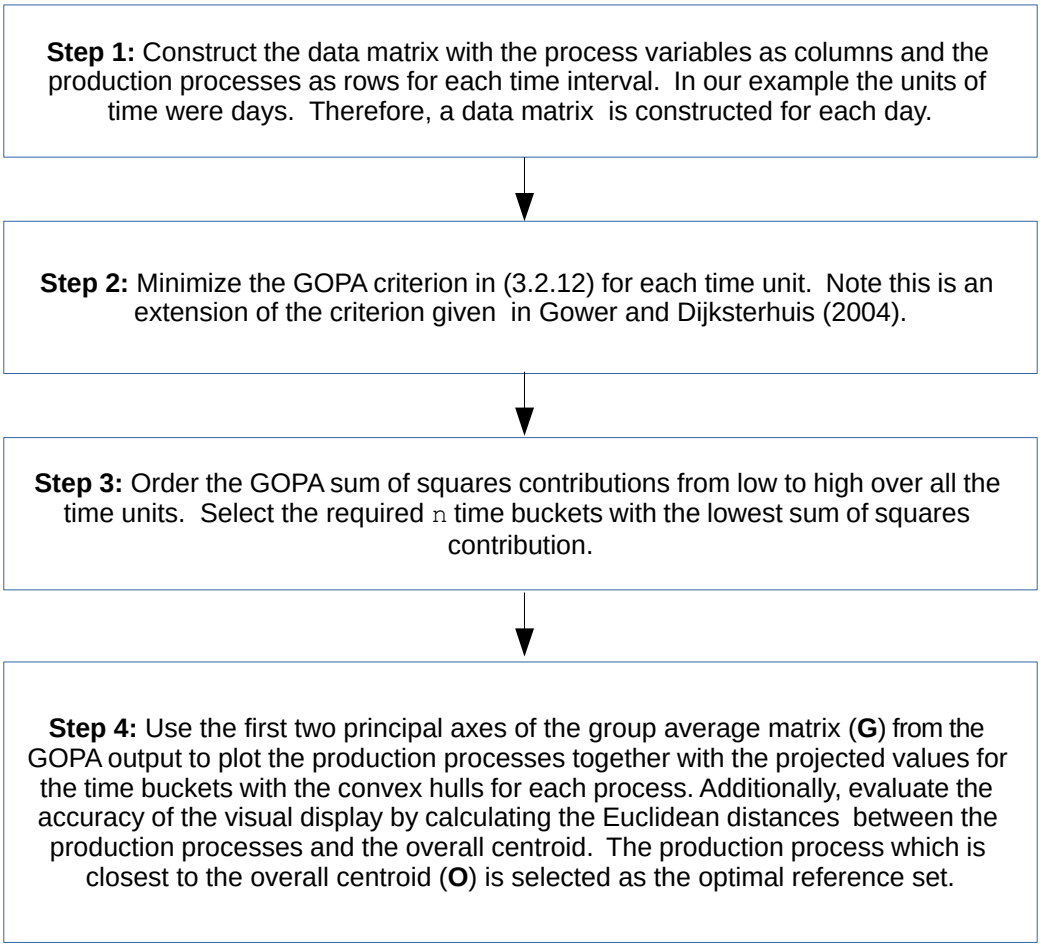


Figure 6.1: Steps for determining the optimal reference set for multivariate monitoring of multiple parallel processes

In Section 3.3.3 the use of the PCA biplot was proposed for short term real-time multivariate process monitoring. The use of a permutation test was investigated to select the appropriate number of principal components to retain. Two different measures of fit for the axes were compared:

- Axes mspe values.
- Axis predictivities.

The axis predictivity criterion was proposed to indicate which axes should be included in each biplot. The developed methodology is summarised in Figure 6.2.

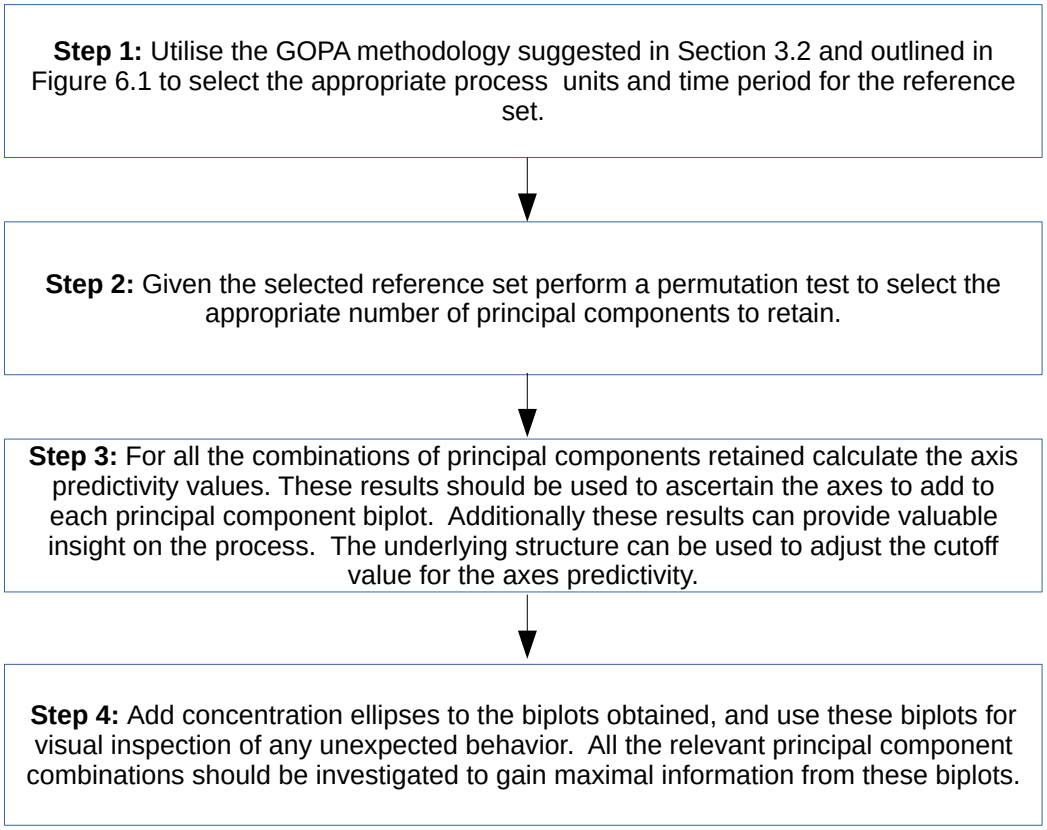


Figure 6.2: Steps for the implementation of a PCA biplot monitoring methodology

In Section 3.4.3 the use of the CVA biplot was proposed for longer term (but real time) monitoring of differences between the gasifiers on a train. The mspe criterion as a measure of fit for the axes proposed by Alves (2012) for PCA biplots was extended to CVA biplots. The proposed criterion was compared to the axis predictivity criterion. It was proposed that the axis predictivity criterion should be used to indicate which axes should be included in the CVA biplots for different combinations of dimensions. The proposed methodology is summarised in Figure 6.3.

In Chapter 4 the focus was on developing and implementing a gasifier performance index (GPI). Three different approaches to a gasifier performance index (GPI) were investigated and compared. The different approaches are:

- *Fundamental GPI* - a purely process driven performance index.
- *Empirical GPI* - a purely data driven performance index.
- *Integrated GPI* - an integrated process and data driven performance index.

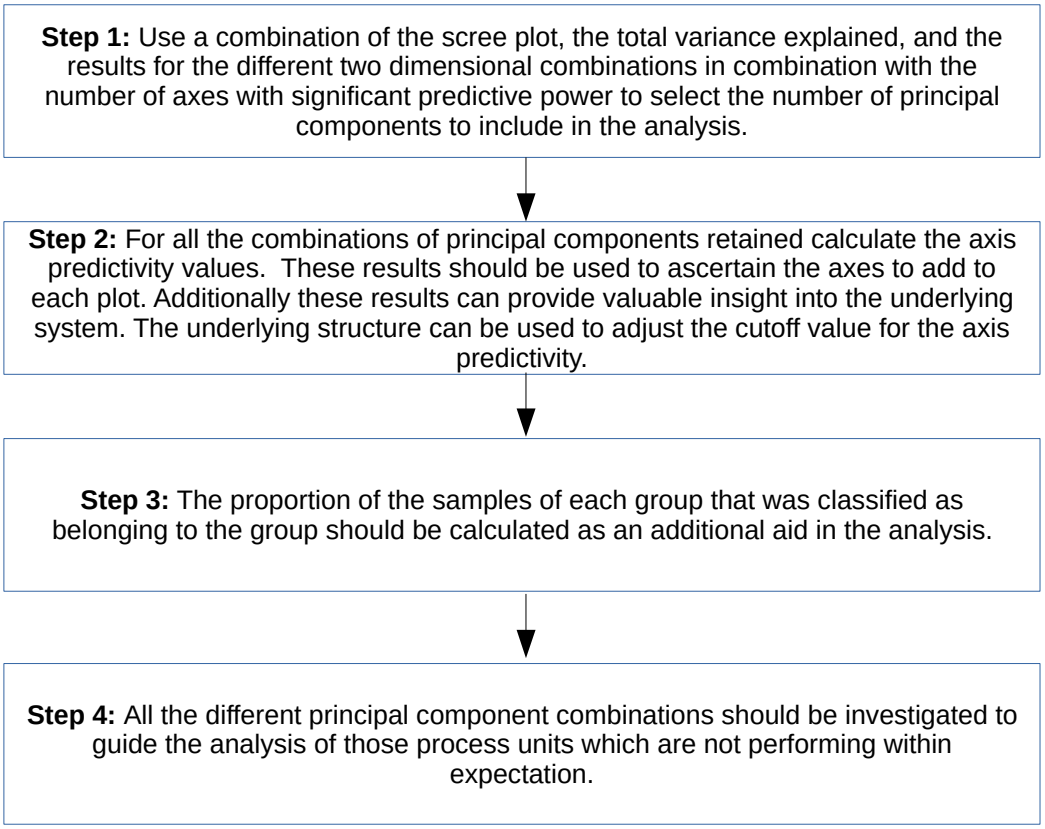


Figure 6.3: Steps for the implementation of a CVA biplot monitoring methodology

The fundamental GPI is based purely on process knowledge and is implemented as a weighted sum of the variables, adjusted by the correction factors and scaled to the same range. The correction factors are related to a factory load variable. The fundamental GPI has several advantageous characteristics. Some of the advantages are:

1. There is no need to define a reference set.
2. The GPI calculation is computationally efficient.
3. The contribution of each variable to the overall GPI is easy to calculate.
4. The GPI calculation is intuitive and easy to relate to the actual production process. This leads to ease of acceptance by the production engineers.

The fundamental approach however possesses two big disadvantages:

1. The GPI value is subjective as all the underlying input values are supplied by subject matter experts.

2. The GPI is by definition univariate, and does not take into account any multivariate relationships between the variables.

In Section 4.2 a novel purely data driven (empirical) approach to the GPI was developed and demonstrated. This index is based on the confidence (α) value at a specified T^2-value. The proposed index gives comparable results to the fundamental GPI values. The methodology is formalised in Figure 6.4. This methodology was proposed as a general data driven performance index as it is objective, and very little prior knowledge of the system is required. Following all the steps in the methodology will lead the user in understanding the underlying process and will provide the interrelationships among the different variables. However, there are some disadvantages to this methodology:

- All the variables are equally weighted.
- Both high and low deviations are equally weighted for the variables.
- A reference data set is required. There is however some advantages in going through the process of obtaining the reference set as valuable knowledge of the underlying process is gathered.
- The empirical GPI is more computationally intensive than the fundamental GPI. However, the mathematical calculations can be stored and do not need to be calculated in real time.

The integrated GPI retains advantages from both the fundamental and empirical GPI, and eliminates some of the major disadvantages. The integrated GPI is calculated by first transforming the data to the offsets from the recommended values scaled by the appropriately specified delta values for the minimum and maximum values. In addition, the correction factor for the factor load variable is applied. Then the empirical GPI algorithm is employed as depicted in Figure 6.4. The integrated GPI therefore combines the scaling, weighting and load adjustments of the fundamental GPI while taking into account the multivariate relationship of the variables similar to the empirical GPI.

The integrated GPI methodology was implemented and compared to the fundamental GPI in Section 4.4. It was concluded that the performance of the integrated GPI was superior to the fundamental GPI. The empirical GPI was not included in the comparison as it could not be used under low factory load conditions. This is a major disadvantage and excludes it for practical implementation.

These results can be generalized to any industrial process as follows:

- If the process knowledge is available:

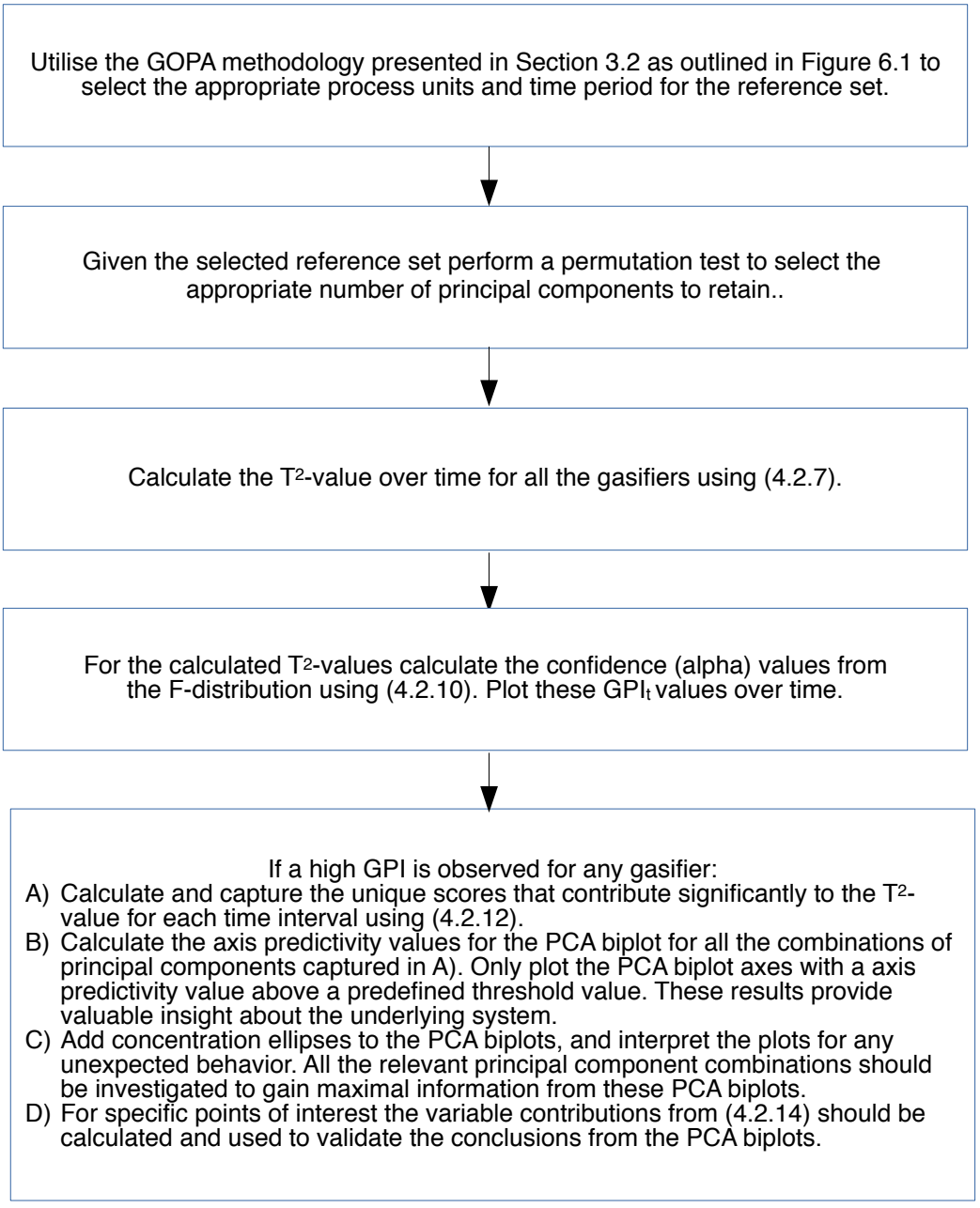


Figure 6.4: Proposed methodology for an empirical performance index

1. Implement and test the integrated GPI in parallel with the fundamental GPI to gain the support of the plant engineers. Deviations between the two approaches can lead to more insight and adjustments to the integrated approach. The recommended values should be monitored and adjusted periodically by utilizing the information from the data driven approach. In addition, the reference set should be reevaluated periodically to ensure the optimal operating conditions have not changed.

- If the process knowledge is not available:

1. Implement the empirical approach as it has many inherent advantageous and will lead to an in-depth understanding of the process and the data. If control variables exist, use the data driven approach to find the underlying effect of the control variables on the dependent variables (see Section 4.2). Use the data to capture the normal operating range and the delta values for the minimum and maximum values.
2. Utilising the knowledge gained from the empirical approach, implement the integrated approach (and possibly for validation the fundamental approach) in parallel and improve until the integrated approach performance meets the requirements. This will not only lead to a better performance index, but also the capturing of the process knowledge that was not available initially.

Chapters 1 to 4 focused on the statistical methodologies implemented and employed for the real-time multivariate MSPEM™ application. In Chapter 5 the software development required to implement the real-time multivariate application was presented. A high level overview of the infrastructure as implemented is provided in Figure 6.5.

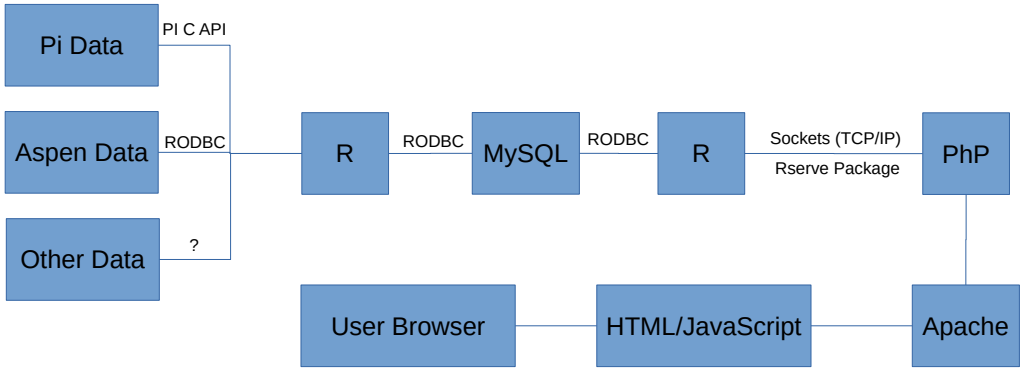


Figure 6.5: MSPEM™ Software Infrastructure Overview

In Sections 5.1 to 5.5 the data interface design decisions and functions were discussed. These functions are responsible for extracting the data from the DCS systems, storing the data in a local database, and providing the interface of the stored data to the statistical functions. Therefore, these functions and interfaces are the fundamental building blocks that make the real-time process monitoring possible.

In Section 5.6 the topic of statistical programming was discussed, and more specifically, the topic of multivariate graphics. Various technologies and design decisions were discussed. The `mltv` package developed for creating multivariate graphics was then discussed in detail. This package is under constant development, and will in future be made available via CRAN or github.

Finally, in Section 5.7 the user interface to the developed MSPEM™ applications was presented. The advantages and disadvantages of a web based interface were discussed, as well as the use of PHP as dynamic web programming language. The MSPEM™ websites for the Sasol Coal Supply and Sasol Gasification applications were illustrated via various screen grabs from the websites.

Underlying the statistical results presented in this study is the need for real time efficient access to various data sources, and the integration of these data sources. This aspect of statistics provides an enormous challenge for the statistician if not designed and implemented in an efficient and standardised manner. Only after the efficient access to integrated and standardised data has been resolved can the statistician apply his skills for the extraction of insight from the data. In addition, the presentation of results to the user is often partly to blame for the lack of acceptance of statistical methodology in industry. Therefore, it is of the utmost importance to strive for a clear, intuitive and visually appealing presentation of the results to foster an adoption of statistical methodology in industry. The current solution to these challenges has been presented. However, although these results serve as a foundation for future work, continued research and development are required to fully resolve these challenges. Significant software development efforts were required to implement the complete real-time multivariate monitoring application.

The work discussed in the current study originated as an investigation into the use of multivariate statistical techniques (specifically biplots) in combination with an integrated, efficient and standardised data management strategy and a web based interface as a viable solution for real-time on-line multivariate process monitoring. The first implementation was deployed on a Linux workstation on the developer's desk. However, the application soon became a business critical application, and had to be migrated to a managed solution. The current architecture is shown in Figure 6.6. The web applications are hosted on two IBM 24 core servers with 64Gb of ram each running Centos 64bit Linux in the IM (Sasol Information Management) data centres (one in Sasolburg and one in Secunda). These servers are jointly administered by Sasol IM, and the Industrial Statistics group. In addition, two Windows vir-

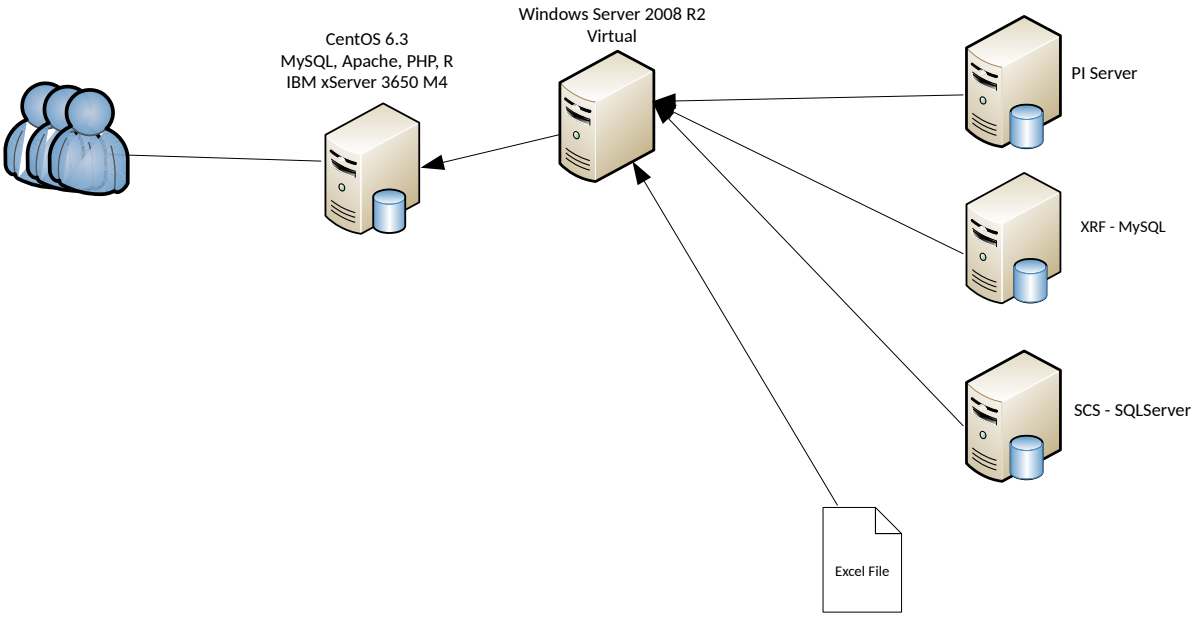


Figure 6.6: MSPEM™ Architecture Overview

tual machines are utilised to populate the local databases from various DCS systems.

The overall multivariate methodology was trademarked by Sasol as the MSPEM™ Technology Package. In addition, following the successful application discussed in this study, various other business units requested MSPEM™ implementations. Currently, the MSPEM™ multivariate process monitoring methodology is implemented or being developed for various Sasol business units including:

- Sasol Synfuels CVC and Gasification.
- Sasol Coal Supply.
- Sasol Synfuels Dashboard.
- Sasol Synfuels Refinery.
- Sasol Synfuels Gas Circuit.
- Sasol Solvents.
- Sasol Waxes.
- Sasol Polymers.

As a direct consequence of the developed modules and implementation discussed in Chapter 2, Sasol has ordered seven additional XRF analysers. These analysers will be installed on the remaining five stackers, as well as on the conveyors feeding the reclaimed coal to the Sasol Gasification plant (one each for the Western and Eastern factories). In addition, a study is in progress to evaluate various technologies to reliably monitor the position of the stackers and reclaimers. As discussed in Section 2.4.2, knowing the position of the stacker will improve the accuracy of the simulation model.

In conclusion, evidence is provided in this study of the successful implementation of the newly developed MSPEM™ real-time multivariate process monitoring methodology. Furthermore, the value added by introducing this facility is widely appreciated.

6.2 Future Research

Various challenges were encountered and addressed in the implementation of this methodology as a critical business application. Future research will focus on:

- Further refinement of the data interface functions to include more process specific information about each variable. This will include, for example, type of variable i.e., temperature, pressure, flow, and standardised units. Including the variable types will allow for the creation of flowsheet interfaces. In addition, this information can be used for structured equation models, as well as mass balance algorithms.

- The integration of interactive JavaScript libraries with the current developed SVG output from the `mltv` package. One of the challenges that will need to be addressed is making this functionality accessible to developers that do not necessarily have a programming background.

- Even though points on a biplot can be labelled it remains difficult to get a clear representation of time. This is especially true for the monitoring of long term trends as the sheer number of points on the biplot can make it virtually impossible to distinguish between the different labels on the plot. The use of SVG animation could alleviate this problem, as the passage of time could be simulated on the biplot by the arrival of points in sequence.

- The integration of additional multivariate statistical techniques such as PLS, PLS-DA and structured equation modelling, into the current methodology. The complexity of the industrial problems frequently necessitates the development of new statistical and data handling methodologies.

- The field of data visualisation is currently receiving a lot of attention due to the boom in Big Data and Data Science. Developing appropriate online graphical visualisation techniques is a challenging research problem that must be addressed to successfully empower users to gain a thorough understanding of the inherent information contained in their data via the multivariate techniques.

- The effect of different desirability functions on the GPI should be investigated. This research should include the selection of optimal weighting and scaling parameters for the desirability functions.

Appendices

- **Appendix A** - Data Interface Software.
- **Appendix B** - Statistical Software.
- **Appendix C** - User Interface.

Appendix A

Data Interface Software

A.1 PI API interface functions (`sslpiutils` and `sslgpipi`)

Listing A.1: `getpidata` C function

```
#include "getpiarc.h"

int32 getpidata(const char *tagname, const char *starttime, const
    char *endtime, long *count, double **pdata, double **pdates,
    long **pstat, TCHAR **emsg, size_t *mcount) {

    int32 lngerror, pipt, prsstd, prsetd, ival, istat, i,
        timestamp, lermsg, tcount, picount, vindex, tmpcount;
    int16 iflag;
    float64 dval, timefrac;
    PIvaluetype ptype;
    TCHAR *efilt;

    char *tgn, *std, *etd, bval[255];

    PITIMESTAMP pistd, pietd, pietd1, pistd1;
    uint32 sbval;

    tgn = malloc((strlen(tagname) + 1) * sizeof (char));
    std = malloc((strlen(starttime) + 1) * sizeof (char));
    etd = malloc((strlen(endtime) + 1) * sizeof (char));
    mcount = malloc(sizeof (size_t));
    strcpy(tgn, tagname);
    strcpy(std, starttime);
    strcpy(etd, endtime);

    lngerror = pipt_findpoint(tgn, &pipt);
    if (lngerror != 0) {
        efilt = malloc((strlen("pipt_findpoint") + 1) * sizeof (
            TCHAR));
        strcpy(efilt, "pipt_findpoint");
```

```
28          lermsg = getpierror(lngerror, *emsg, mcount, efilt);
29          free(efilt);
30          free(tgn);
31          free(std);
32          free(etd);
33          return (lngerror);
34
35      }
36
37      lngerror = pipt_pointtypex(pipt, &ptype);
38      if (lngerror != 0) {
39          efilt = malloc((strlen("pipt_pointtypex") + 1) * sizeof (
                TCHAR));
40          strcpy(efilt, "pipt_pointtypex");
41          lermsg = getpierror(lngerror, *emsg, mcount, efilt);
42          free(efilt);
43          free(tgn);
44          free(std);
45          free(etd);
46          return (lngerror);
47      }
48      lngerror = pitm_parsetime(std, 0, &prsstd);
49      lngerror = pitm_parsetime(etd, 0, &prsetd);
50
51      lngerror = pitm_setpitime(&pistd, prsstd, 0);
52      lngerror = pitm_setpitime(&pistd1, prsstd, 0);
53      lngerror = pitm_setpitime(&pietd, prsetd, 0);
54      lngerror = pitm_setpitime(&pietd1, prsetd, 0);
55      tcount = 0;
56      //Start the index at 1 because I want to save the "before"
            value in 0
57      vindex = 1;
58      picount = *count;
59      tmpcount = *count;
60
61      lngerror = piar_getarcvaluesx(pipt, ARCflag_comp, &picount, &
            dval, &ival, &bval, &sbval, &istat, &iflag, &pistd, &pietd,
             GETFIRST);
62      if (lngerror == 0) {
63          tcount += picount;
64
65          if (picount > 0) {
66              (*pdata) = (double*) malloc((picount + 2) * sizeof (
                    double));
67              (*pdates) = (double*) malloc((picount + 2) * sizeof (
                    double));
68              (*pstat) = (long*) malloc((picount + 2) * sizeof (long
                    ));
69              (*pdata)[vindex] = dval;
70              timestamp = pitm_getpitime(&pietd, &timefrac);
71              (*pdates)[vindex] = timestamp + timefrac;
72              (*pstat)[vindex] = istat;
```

```
73              vindex++;
74
75              while (picount > 0) {
76                  for (i = 1; i < picount; i++) {
77                      lngerror = piar_getarcvaluesx(pipt, ARCflag_
                            comp, &tmpcount, &dval, &ival, &bval, &
                            sbval, &istat, &iflag, &pistd, &pietd,
                            GETNEXT);
78                      (*pdata)[vindex] = dval;
79                      timestamp = pitm_getpitime(&pietd, &timefrac);
80                      (*pdates)[vindex] = timestamp + timefrac;
81                      (*pstat)[vindex] = istat;
82                      vindex++;
83                  }
84
85                  //More than *count values available
86                  if (picount == *count) {
87
88                      pistd = pietd;
89                      pietd = pietd1;
90                      lngerror = piar_getarcvaluesx(pipt, ARCflag_
                            comp, &picount, &dval, &ival, &bval, &sbval
                            , &istat, &iflag, &pistd, &pietd, GETFIRST)
                            ;
91                      if (picount > 0) {
92                          tcount += (picount - 1);
93                          (*pdata) = (double*) realloc(*pdata, (
                                tcount + 2) * sizeof (double));
94                          (*pdates) = (double*) realloc(*pdates, (
                                tcount + 2) * sizeof (double));
95                          (*pstat) = (long*) realloc(*pstat, (tcount
                                + 2) * sizeof (long));
96                      }
97                  } else {
98                      picount = 0;
99                  }
100             }
101
102             lngerror = piar_getarcvaluex(pipt, ARCVALUEAFTER, &
                    dval, &ival, &bval, &sbval, &istat, &iflag, &pietd1
                    );
103             (*pdata)[tcount + 1] = dval;
104             timestamp = pitm_getpitime(&pietd1, &timefrac);
105             (*pdates)[tcount + 1] = timestamp + timefrac;
106             (*pstat)[tcount + 1] = istat;
107
108             lngerror = piar_getarcvaluex(pipt, ARCVALUEBEFORE, &
                    dval, &ival, &bval, &sbval, &istat, &iflag, &pistd1
                    );
109             (*pdata)[0] = dval;
110             timestamp = pitm_getpitime(&pistd1, &timefrac);
111             (*pdates)[0] = timestamp + timefrac;
```

```
112             (*pstat)[0] = istat;
113             *count = tcount + 2;
114         }
115     } else {
116
117         if (lngerror == 100) {
118             (*pdata) = (double*) malloc((2) * sizeof (double));
119             (*pdates) = (double*) malloc((2) * sizeof (double));
120             (*pstat) = (long*) malloc((2) * sizeof (long));
121             lngerror = piar_getarcvaluex(pipt, ARCVALUEAFTER, &
                    dval, &ival, &bval, &sbval, &istat, &iflag, &pietd1
                    );
122             (*pdata)[1] = dval;
123             timestamp = pitm_getpitime(&pietd1, &timefrac);
124             (*pdates)[1] = timestamp + timefrac;
125             (*pstat)[1] = istat;
126
127             lngerror = piar_getarcvaluex(pipt, ARCVALUEBEFORE, &
                    dval, &ival, &bval, &sbval, &istat, &iflag, &pistd1
                    );
128             (*pdata)[0] = dval;
129             timestamp = pitm_getpitime(&pistd1, &timefrac);
130             (*pdates)[0] = timestamp + timefrac;
131             (*pstat)[0] = istat;
132             *count = tcount + 2;
133
134         } else {
135             efilt = malloc((strlen("piar_getarcvaluesx") + 1) *
                    sizeof (TCHAR));
136             strcpy(efilt, "piar_getarcvaluesx");
137             lermsg = getpierror(lngerror, *emsg, mcount, efilt);
138             free(efilt);
139         }
140
141     }
142     free(tgn);
143     free(std);
144     free(etd);
145     free(mcount);
146
147     return lngerror;
148 }
```

Listing A.2: `imppidata` C function

```
1
2  #ifndef PIUTIL_C
3  #define PIUTIL_C
4
5  #include <windows.h>
6  #include <R.h>
7  #include <Rdefines.h>
8  #include <Rinternals.h>
9  #include <stdio.h>
10 #include <math.h>
11 #include "./PIInclude/piapi.h"
12 #include "./PIInclude/piapix.h"
13 #include "getpiarc.h"
14
15 SEXP imppidata(SEXP tagnames, SEXP bdate, SEXP edate, SEXP ntags)
       {
16   double **pidat, **pidate;
17   long *picount, maxcount, i, j, nvtgs, **pistat;
18   int *RPiCount, *ntgs = INTEGER(ntags);
19   long *RPiStat;
20   SEXP PiData, PiDates, PiCounts, ans, PiStat,Ermsg, ListNames;
21   double *RPiData, *RPiDates;
22   const char **tgn;
23   const char *cbdate = CHAR(STRING_ELT(bdate, 0));
24   const char *cedate = CHAR(STRING_ELT(edate, 0));
25   int32 lngerror,lermsg;
26   size_t mcount;
27  TCHAR *emsg;
28
29   mcount = 5024;
30   emsg = malloc((mcount) * sizeof (TCHAR));
31   strcpy(emsg,"Success");
32   lngerror = 0;
33   Rprintf("%s \n", emsg);
34   if(lngerror == 0) {
35
36     tgn = malloc((*ntgs) * sizeof (char*));
37     for (i = 0; i < (*ntgs); i++) {
38       tgn[i] = CHAR(STRING_ELT(tagnames, i));
39     }
40
41     pidat = malloc((*ntgs) * sizeof (double*));
42     pidate = malloc((*ntgs) * sizeof (double*));
43     pistat = malloc((*ntgs) * sizeof (long*));
44     picount = malloc((*ntgs) * sizeof (long));
45     nvtgs = 0;
46     for (i = 0; i < (*ntgs); i++) {
47       Rprintf("%s \n",tgn[i]);
48       picount[i] = 100000;
49       lngerror = getpidata(tgn[i], cbdate, cedate, &picount[i], &
             pidat[i], &pidate[i],&pistat[i],&emsg,&mcount);
```

```
50
51        Rprintf("lng: %d %d \n",lngerror,picount[i]);
52        if(lngerror != 0) {
53    strcat(emsg," : ");
54    strcat(emsg,tgn[i]);
55    break;
56        }
57        nvtgs++;
58      }
59      if(lngerror == 0) {
60        maxcount = 0;
61        for (i = 0; i < (*ntgs); i++) {
62          Rprintf("Picount %d = %d \n",i,picount[i]);
63          Rprintf("PiDate: %f %f \n",pidat[i][picount[i]-1],pidate[i
                ][picount[i]-1]);
64          if (maxcount < picount[i])
65      maxcount = picount[i];
66
67        }
68        PROTECT(PiData = allocMatrix(REALSXP, maxcount, (*ntgs)));
69        PROTECT(PiDates = allocMatrix(REALSXP, maxcount, (*ntgs)));
70        PROTECT(PiCounts = allocVector(INTSXP, (*ntgs)));
71        PROTECT(PiStat = allocMatrix(INTSXP, maxcount,(*ntgs)));
72
73        RPiData = REAL(PiData);
74        RPiDates = REAL(PiDates);
75        RPiCount = INTEGER(PiCounts);
76        RPiStat = INTEGER(PiStat);
77
78        for (j = 0; j < *(ntgs); j++) {
79          RPiCount[j] = picount[j];
80          for (i = 0; i < picount[j]; i++) {
81      RPiData[i + j * (maxcount)] = pidat[j][i];
82      RPiDates[i + j * maxcount] = pidate[j][i];
83      RPiStat[i + j * maxcount] = pistat[j][i];
84          }
85          free(pidat[j]);
86          free(pidate[j]);
87          free(pistat[j]);
88        }
89        free(pidat);
90        free(pidate);
91        free(pistat);
92        free(picount);
93        free(tgn);
94        lngerror = pilogout();
95        PROTECT(ans = allocVector(VECSXP, 5));
96
97        SET_VECTOR_ELT(ans, 0, PiData);
98        SET_VECTOR_ELT(ans, 1, PiDates);
99        SET_VECTOR_ELT(ans, 2, PiCounts);
100       SET_VECTOR_ELT(ans, 3, PiStat);
```

```
101
102
103         PROTECT(Ermsg=allocVector(STRSXP,1));
104         SET_STRING_ELT(Ermsg,0,mkChar(emsg));
105         SET_VECTOR_ELT(ans,4,Ermsg);
106
107         PROTECT(ListNames = allocVector(STRSXP,5));
108
109         SET_STRING_ELT(ListNames,0,mkChar("PiData"));
110         SET_STRING_ELT(ListNames,1,mkChar("PiDates"));
111         SET_STRING_ELT(ListNames,2,mkChar("PiCounts"));
112         SET_STRING_ELT(ListNames,3,mkChar("PiStat"));
113         SET_STRING_ELT(ListNames,4,mkChar("Ermsg"));
114
115         setAttrib(ans,R_NamesSymbol,ListNames);
116         UNPROTECT(7);
117         return ans;
118       }
119       for (j = 0; j <= nvtgs; j++) {
120         free(pidat[j]);
121         free(pidate[j]);
122         free(pistat[j]);
123       }
124       free(picount);
125       free(pidat);
126       free(pidate);
127       free(pistat);
128       free(tgn);
129     }
130 lngerror = pilogout();
131     //This should only run when no error occured
132     PROTECT(ans = allocVector(VECSXP, 1));
133     PROTECT(Ermsg=allocVector(STRSXP,1));
134     PROTECT(ListNames = allocVector(STRSXP,1));
135     SET_STRING_ELT(Ermsg,0,mkChar(emsg));
136     SET_VECTOR_ELT(ans,0,Ermsg);
137     SET_STRING_ELT(ListNames,0,mkChar("Ermsg"));
138     setAttrib(ans,R_NamesSymbol,ListNames);
139     free(emsg);
140     UNPROTECT(3);
141
142     return ans;
143 }
144
145
146 #endif          //  #ifndef PIUTIL_C
```

Listing A.3: `sslpiutils::imppidata` function

```
#' Function to download PI Data between two dates
#'
#' This function downloads data from PI through the PI C API
#'
#' @param tagnames Tags to download
#' @param bdate Start date to download from
#' @param edate End date for data download
#'
#' @return
#' Returns a list with four items
#' \item{PiData}{A Matrix of data values with a column per tag.}
#' \item{PiDates}{A Matrix of datetime values with a column per tag.}
#' \item{PiStat}{A Matrix of status values with a column per tag.}
#' \item{Ermsg}{A status value from the PI API. Either 'Success' or
#' a error value.}
#'
#'
#' @export
#'

imppidata <- function(tagnames,bdate,edate) {
    pidata <- .Call("imppidata",tagnames,bdate,edate,length(tagnames))
    ##Time discrepancy between PI and R. PI uses UTC and R uses SAST which is UTC + 2.
    print(pidata$Ermsg)
    if(pidata$Ermsg == "Success") {
        pidata[[2]] <- pidata[[2]]-2*3600

        for(i in 1:length(pidata$PiCounts))
        {
            if(pidata$PiCounts[i]<nrow(pidata$PiDates)){
                pidata$PiDates[(pidata$PiCounts[i]+1):nrow(pidata$PiDates),i] <- 0
                pidata$PiStat[(pidata$PiCounts[i]+1):nrow(pidata$PiStat),i] <- 0

            }

        }
    }
  return(pidata)
}
```

Listing A.4: `getcurpival` C function

```
#include "getpiarc.h"


int32 getcurpival(const char *tagname, const char *curtime, double
    **pdata, double **pdates, long **pstat, TCHAR **emsg, size_t *
    mcount) {

    int32 lngerror, pipt, prsstd, ival, istat, timestamp, lermsg;

    int16 iflag;
    float64 dval, timefrac;
    PIvaluetype ptype;
    TCHAR *efilt;
    char *tgn, *std, bval[255];

    PITIMESTAMP pcurtd;
    uint32 sbval;

    tgn = malloc((strlen(tagname) + 1) * sizeof (char));
    std = malloc((strlen(curtime) + 1) * sizeof (char));

    mcount = malloc(sizeof (size_t));
    strcpy(tgn, tagname);
    strcpy(std, curtime);

    lngerror = pipt_findpoint(tgn, &pipt);
    if (lngerror != 0) {
        *mcount = 5024;
        (*emsg) = malloc((*mcount) * sizeof (TCHAR));
        efilt = malloc((strlen("pipt_findpoint") + 1) * sizeof (
            TCHAR));
        strcpy(efilt, "pipt_findpoint");
        lermsg = getpierror(lngerror, *emsg, mcount, efilt);
        free(efilt);
        free(tgn);
        free(std);

        return (lngerror);

    }
    lngerror = pipt_pointtypex(pipt, &ptype);
    if (lngerror != 0) {
        *mcount = 5024;
        (*emsg) = malloc((*mcount) * sizeof (TCHAR));
        efilt = malloc((strlen("pipt_pointtypex") + 1) * sizeof (
            TCHAR));
        strcpy(efilt, "pipt_pointtypex");
        lermsg = getpierror(lngerror, *emsg, mcount, efilt);
        free(efilt);
        free(tgn);
        free(std);
```

```
48          return (lngerror);
49      }
50
51      lngerror = pitm_parsetime(std, 0, &prsstd);
52      lngerror = pitm_setpitime(&pcurtd, prsstd, 0);
53
54      lngerror = piar_getarcvaluex(pipt, ARCVALUEBEFORE, &dval, &
            ival, &bval, &sbval, &istat, &iflag, &pcurtd);
55      if (lngerror != 0) {
56           *mcount = 5024;
57          (*emsg) = malloc((*mcount) * sizeof (TCHAR));
58          efilt = malloc((strlen("piar_getarcvaluex") + 1) * sizeof
                (TCHAR));
59          strcpy(efilt, "piar_getarcvaluex");
60          lermsg = getpierror(lngerror, *emsg, mcount, efilt);
61          free(efilt);
62          free(tgn);
63          free(std);
64          return (lngerror);
65      }
66      (*pdata) = (double*) malloc(sizeof (double));
67      (*pdates) = (double*) malloc(sizeof (double));
68      (*pstat) = (long*) malloc(sizeof (long));
69      (*pdata)[0] = dval;
70      timestamp = pitm_getpitime(&pcurtd, &timefrac);
71      (*pdates)[0] = timestamp + timefrac;
72      (*pstat)[0] = istat;
73      free(tgn);
74      free(std);
75
76      return lngerror;
77  }
```

Listing A.5: `impcurpival` C function

```
1
2  #ifndef PIUTIL_C
3  #define PIUTIL_C
4
5  #include <windows.h>
6  #include<R.h>
7  #include<Rdefines.h>
8  #include<Rinternals.h>
9  #include<stdio.h>
10 #include<math.h>
11 #include "./PIInclude/piapi.h"
12 #include "./PIInclude/piapix.h"
13 #include "getpiarc.h"
14
15 SEXP impcurpival(SEXP tagnames, SEXP cdate, SEXP ntags) {
16   double **pidat, **pidate;
17   long  i, j, **pistat, nvtgs;
18   int  *ntgs = INTEGER(ntags);
19   long *RPiStat;
20   SEXP PiData, PiDates, ans, PiStat, ListNames, Ermsg;
21   double *RPiData, *RPiDates;
22   const char **tgn;
23   const char *cbdate = CHAR(STRING_ELT(cdate, 0));
24   size_t mcount;
25   TCHAR *emsg;
26
27   int32 lngerror;
28
29   mcount = 5024;
30   emsg = malloc((mcount) * sizeof (TCHAR));
31   strcpy(emsg,"Success");
32   lngerror = pilogin(&emsg,&mcount);
33   Rprintf("%d %s \n",lngerror,emsg);
34   if(lngerror==0) {
35     tgn = malloc((*ntgs) * sizeof (char*));
36     for (i = 0; i < (*ntgs); i++) {
37       tgn[i] = CHAR(STRING_ELT(tagnames, i));
38     }
39
40     pidat = malloc((*ntgs) * sizeof (double*));
41     pidate = malloc((*ntgs) * sizeof (double*));
42     pistat = malloc((*ntgs) * sizeof (long*));
43     nvtgs = 0;
44     for (i = 0; i < (*ntgs); i++) {
45       Rprintf("%s \n",tgn[i]);
46       lngerror = getcurpival(tgn[i], cbdate, &pidat[i],
47           &pidate[i],&pistat[i],&emsg,&mcount);
48       Rprintf("lng: %d %d \n",lngerror);
49       if(lngerror != 0) {
50   strcat(emsg," : ");
51   strcat(emsg,tgn[i]);
```

```
52    Rprintf("lng: %s \n",emsg);
53    break;
54        }
55        nvtgs++;
56      }
57      if(lngerror == 0) {
58      PROTECT(PiData = allocMatrix(REALSXP, 1, (*ntgs)));
59      PROTECT(PiDates = allocMatrix(REALSXP, 1, (*ntgs)));
60      PROTECT(PiStat = allocMatrix(INTSXP, 1,(*ntgs)));
61      PROTECT(ans = allocVector(VECSXP, 4));
62      RPiData = REAL(PiData);
63      RPiDates = REAL(PiDates);
64      RPiStat = INTEGER(PiStat);
65
66      for (j = 0; j < *(ntgs); j++) {
67
68        RPiData[j] = pidat[j][0];
69        RPiDates[j] = pidate[j][0];
70        RPiStat[j] = pistat[j][0];
71
72        free(pidat[j]);
73        free(pidate[j]);
74        free(pistat[j]);
75      }
76      free(pidat);
77      free(pidate);
78      free(pistat);
79
80      SET_VECTOR_ELT(ans, 0, PiData);
81      SET_VECTOR_ELT(ans, 1, PiDates);
82      SET_VECTOR_ELT(ans, 2, PiStat);
83      PROTECT(Ermsg=allocVector(STRSXP,1));
84      SET_STRING_ELT(Ermsg,0,mkChar(emsg));
85      SET_VECTOR_ELT(ans,3,Ermsg);
86
87      PROTECT(ListNames = allocVector(STRSXP,4));
88
89      SET_STRING_ELT(ListNames,0,mkChar("PiData"));
90      SET_STRING_ELT(ListNames,1,mkChar("PiDates"));
91      SET_STRING_ELT(ListNames,2,mkChar("PiStat"));
92      SET_STRING_ELT(ListNames,3,mkChar("Ermsg"));
93      setAttrib(ans,R_NamesSymbol,ListNames);
94
95      UNPROTECT(6);
96      Rprintf("%d %s \n",lngerror,emsg);
97
98      return ans;
99    }
100     for(j = 0; j <= nvtgs; j++) {
101       free(pidat[j]);
102       free(pidate[j]);
103     }
```

```
104     }
105     //This should only run when an error occured
106     PROTECT(ans = allocVector(VECSXP, 1));
107     PROTECT(Ermsg=allocVector(STRSXP,1));
108     PROTECT(ListNames = allocVector(STRSXP,1));
109     SET_STRING_ELT(Ermsg,0,mkChar(emsg));
110     SET_VECTOR_ELT(ans,0,Ermsg);
111     SET_STRING_ELT(ListNames,0,mkChar("Ermsg"));
112     setAttrib(ans,R_NamesSymbol,ListNames);
113     free(emsg);
114     UNPROTECT(3);
115
116     return ans;
117 }
118
119 #endif          //  #ifndef PIUTIL_C
```

Listing A.6: `sslpiutils::impcurpival` function

```
1  #' Function to download current PI Value
2  #'
3  #' This function downloads current PI values for the tags
4  #'
5  #' @param tagnames Tags to download
6  #' @param cdate Current time value
7  #'
8  #'
9  #' @return
10 #' Returns a list with four items
11 #' \item{PiData}{Current Data Values for Tags}
12 #' \item{PiDates}{Timestamp values for the Tags.}
13 #' \item{PiStat}{Status values for the Tags.}
14 #' \item{Ermsg}{A status value from the PI API. Either 'Success' or
15 #' a error value.}
16 #'
17 #'
18 #' @export
19 #'
20
21 impcurpival <- function(tagnames,cdate=getcurrtime(format="%d-%b-%y %H:%M:%S")) {
22     pidata <- .Call("impcurpival",tagnames,cdate,length(tagnames))
23     ##Time discrepancy between PI and R. PI uses UTC and R uses SAST which is UTC + 2.
24     ##print(pidata)
25     if(pidata$Ermsg == "Success") {
26         pidata[[2]] <- pidata[[2]]-2*3600
27     }
28     return(pidata)
29 }
```

Listing A.7: `sslpiutils::buildtags` function

```
#' Function to create and initialize Tag Tables
#'
#' This function creates the tag tables, and thereafter initialize the tables with
#' data. The input for buildtags are in Table \code{TagStatus}.
#'
#' @param dbname Name of database to export to
#' @param dbtype Type of database to export to
#' @param datetime Column with datetime values
#' @param schemaname Schema in database where tables exist
#'
#' @export
#'


buildtags <- function(dbname,dbtype,datetime,schemaname) {
    options(stringsAsFactors=FALSE)
    ad <- dbconn(dbname,dbtype)
    if(!is.null(ad)) {
        TagStatus <- sqlQuery(ad,"SELECT * FROM `TagStatus`")
        ## Keep TAGS that either does not exist yet, or have not been populated yet
        TagStatus <- TagStatus[rowSums(TagStatus[,c("CREATED","POPULATED")])< 2 ,,drop=FALSE]
        if(nrow(TagStatus)>0) {
            for(i in 1:nrow(TagStatus)) {
                if(TagStatus[i,"CREATED"]==0) {
                    sqltmp <- gsub("PLH",TagStatus[i,"TAGNAME"],sqltemplate)
                    res <- sqlQuery(ad,sqltmp)
                    tblxst <- tblexist(dbname,dbtype,TagStatus[i,"TAGNAME"],schemaname)
                    if((res[1] == "No Data")|((!is.null(tblxst)&(tblxst>0)))) {
                        sqlupd <- paste("UPDATE `TagStatus` SET `CREATED` = 1 WHERE `TAGNAME` = '",
                                        TagStatus[i,"TAGNAME"],"';",sep="")
                        res <- sqlQuery(ad,sqlupd)
                    }
                }
                tblxst <- tblexist(dbname,dbtype,TagStatus[i,"TAGNAME"],schemaname)
                if(!is.null(tblxst)&(tblxst>0)) {
                    if((TagStatus[i,"POPULATED"]==0)) {
                        inittag(TagStatus[i,"TAGNAME"],TagStatus[i,"SDATE"],
                                TagStatus[i,"UPDATESTEPS"],dbname,dbtype,datetime)
                        if(tblrows(dbname,dbtype,TagStatus[i,"TAGNAME"],datetime,schemaname)>0){
```

```
40                                  sqlupd <- paste("UPDATE `TagStatus`
                                        SET `POPULATED` = 1 WHERE `TAGNAME`
                                         = '",
41                                                  TagStatus[i,"TAGNAME"
                                                        ],"';",sep="")
42                                  res  <- sqlQuery(ad,sqlupd)
43                                  } else {
44                                      errlog(dbname,dbtype,"LogTable",
                                        appname,"buildtags",
45                                          paste("No data exported to : ",
                                            TagStatus[i,"TAGNAME"],sep="
                                            "))
46                                  }
47                              }
48                          } else {
49                              errlog(dbname,dbtype,"LogTable",appname,"
                                buildtags",
50                                  paste("Failed to create table : ",
                                    TagStatus[i,"TAGNAME"],sep=""))
51                          }
52                      }
53                  }
54              } else {
55                  return(NULL)
56              }
57          }
```

Listing A.8: `sslpiutils::inittag` function

```
#' Function to inititialize Table with Data
#'
#' This function inititializes an empty table with data. This function is typically run
#' immediadly after creating the table from \code{buildtags}. The table is initialized from
#' \code{bdate} with step \code{sizes} (to minimize instantaneous memory use).
#'
#' @param inittag Table name to initialize
#' @param bdate Date to initialize from
#' @param ssize Step size for initialization
#' @param dbname Name of database to export to
#' @param dbtype Type of database to export to
#' @param datetime Column name in database with timestamp values
#'
#' @return Return either number of rows in table \code{inittag} after initialization or NULL if error occured.
#'
#' @export
#'
#


inittag <- function(inittag,bdate,ssize=NULL,dbname,dbtype,datetime) {
 options(stringsAsFactors=FALSE)
 ad <- dbconn(dbname,dbtype)
 if(!is.null(ad)) {

     pcv <- impcurpival(inittag)
     if(pcv$Ermsg=="Success") {


         edate <- dateconvrt(pcv$PiDates[1],format="%d-%b-%y %H:%M:%S")

         if(is.null(ssize)) {
             pidat <- imppidata(inittag,bdate,edate)
             if(pidat$Ermsg=="Success") {
                 pi.data <- cbind(pidat[["PiDates"]],pidat[["PiData"]],pidat[["PiStat"]])
                 colnames(pi.data) <- c(datetime,"Value","Status")
                 pi.data <- pi.data[pi.data[,1]>datestrip(bdate),,drop=FALSE]
                 pi.data <- pi.data[-nrow(pi.data),,drop=FALSE]

                 if(!is.null(pi.data)) {
                     exptodbc(dbname,pi.data,paste("`",inittag,"`",sep=""),dbtype)
```

```
43                      }
44                      close(ad)
45                  } else {
46                      close(ad)
47                      errlog(dbname,dbtype,"LogTable",appname,"
                            inittags",pidat$Ermsg)
48                      return(NULL)
49                  }
50              } else {
51                  ed <- dateconvrt(datestrip(bdate)+ ssize*24*3600,
52                                   format="%d-%b-%y %H:%M:%S")
53                  init <- TRUE
54                  while(datestrip(ed) < datestrip(edate)) {
55                      pidat <- imppidata(inittag,bdate,ed)
56                      if(pidat$Ermsg == "Success") {
57                          pi.data <- cbind(pidat[["PiDates"]],pidat
                                [["PiData"]],pidat[["PiStat"]])
58                          colnames(pi.data)  <- c(datetime,"Value","
                                Status")
59                          if(init) {
60                              pi.data <- pi.data[pi.data[,1]>
                                    datestrip(bdate),,drop=FALSE]
61                          }
62
63                          if(!is.null(pi.data)) {
64                              exptodbc(dbname,pi.data,paste("'",
                                    inittag,"'",sep=""),dbtype)
65                              init <- FALSE
66                              bdate <- dateconvrt(datestrip(
                                    getmaxtime(ad,paste("'",inittag,"'
                                    ,sep=""),datetime)),format="%d-%b-%
                                    y %H:%M:%S")
67                          }
68                          ed <- dateconvrt(datestrip(bdate)+ ssize*
                                24*3600,format="%d-%b-%y %H:%M:%S")
69                          print(bdate)
70                          print(ed)
71                      } else {
72                          close(ad)
73                          errlog(dbname,dbtype,"LogTable",appname,"
                                inittags",pidat$Ermsg)
74                          return(NULL)
75                      }
76                  } ## end while
77              }
78
79              close(ad)
80          } else {
81              close(ad)
82              errlog(dbname,dbtype,"LogTable",appname,"inittags",pcv
                    $Ermsg)
83              return(NULL)
```

```
84          }
85      } else {
86          errlog(dbname,dbtype,"LogTable",appname,"inittags",
87                  paste("Failed to connect to database : ",dbname,sep
                        =""))
88          return(NULL)
89      }
90      return(tblrows(dbname,dbtype,inittag,datetime,schemaname))
91
92  }
```

Listing A.9: `sslpiutils::sqltemplate` data

```
1  sqltemplate <- "CREATE TABLE `PLH` (
2    `DateTime` double NOT NULL,
3    `Value` double DEFAULT NULL,
4    `Status` varchar(45) DEFAULT NULL,
5    PRIMARY KEY (`DateTime`)
6  );"
```

Listing A.10: `sslgpipi::updgpidb` function

```
1  #' Function to update Synfuels Gasification Data
2  #'
3  #' This function updates the tags stored in \code{TagStatus} table
4  #' that are both populated and created. Additionally there exists
       an
5  #' \code{UPDATE} column to override the updates. The updgroup
       variable
6  #' serves as an indicator to split the updates to utilize multiple
        update
7  #' processes. The update groups are set in the \code{TagStatus}
       table.
8  #'
9  #' @param updgroup Which update group to update. \code{NULL} for
       all.
10 #'
11 #' @export
12
13 updgpidb <- function(updgroup=1){
14
15   ad <- dbconn(dbname,dbtype)
16   options(stringsAsFactors=FALSE)
17   TagStatus <- sqlQuery(ad,"SELECT * FROM `TagStatus`;")
18   ##Filter out tags that are not in current update group
19   ##If group is NULL all tag groups will be updated
20   if(!is.null(updgroup)){
21     TagStatus  <- TagStatus[TagStatus[,"UPDATEGROUP"]==updgroup,]
22   }
23   ##Filter out tags that are not active yet
24   TagStatus  <- TagStatus[rowSums(TagStatus[,c("CREATED","
         POPULATED")])==2,]
25   ##Filter out tags that should not be updated
```

```
26    TagStatus  <- TagStatus[TagStatus[,"UPDATE"]==1,]
27
28    ct <- 1
29    tags <- paste("`",TagStatus[,"TAGNAME"],"`",sep="")
30    for(i in tags) {
31        tag <- gsub("`","",i)
32        pcv <- impcurpival(tag)
33        mdbt <- getmaxtime(ad,i,datetime)
34
35        if((pcv$PiDates > datestrip(mdbt))&(!is.na(date.strip(mdbt))
              )) {
36            mdbt <- dateconvrt(datestrip(mdbt),format="%d-%b-%y %H:%
                  M:%S")
37            edate <- dateconvrt(pcv$PiDates[1,ct],format="%d-%b-%y %
                  H:%M:%S")
38            print(i)
39
40            pidat <- imppidata(tag,mdbt,edate)
41            pi.data <- cbind(pidat[["PiDates"]],pidat[["PiData"]],
                  pidat[["PiStat"]])
42            colnames(pi.data)  <- c(datetime,"Value","Status")
43            pi.data <- pi.data[pi.data[,1]>datestrip(mdbt),,drop=
                  FALSE]
44            pi.data <- pi.data[-nrow(pi.data),,drop=FALSE]
45
46            if(!is.null(pi.data)&!is.null(nrow(pi.data))&nrow(pi.
                  data)>0) {
47                exptodbc(pi.data,i,dbname,dbtype)
48            }
49
50        }
51    }
52    sqlstr <- paste("INSERT INTO `lastupd` VALUES (",
53                    datestrip(getcurrtime()),");",sep="")
54    sqlQuery(ad,sqlstr)
55    close(ad)
56 }
```

A.2 Date Conversion Functions (ssldtutils)

Listing A.11: ssldtutils::dateconvrt function

```
##' Function to convert numeric date values to string date values
##'
##' Utility function to convert numeric date values (in the POSiX
    format) to a string date
##'
##' @param idate : Numeric date in seconds from origin.
##' @param origin : Origin for numeric date
##' @param format : Format for string date (see ?strptime)
##'
##' @return Date in string format
##'
##' @seealso datestrip
##'
##' @export

dateconvrt <-
  function(idate,origin="1970-01-01",format="%d-%b-%y %H:%M:%OS1")
  {
    date.convrt <- format(as.POSIXlt(idate,origin=origin),format=
        format)
    return(date.convrt)
  }
```

Listing A.12: ssldtutils::datestrip function

```
##' Function to convert string  date values to numeric date values
##'
##' Utility function to convert string date values to a numeric
    date (in the POSiX format)
##'
##' @param sdate : Date in string format.
##' @param format : Format of string date (see ?strptime)
##'
##' @return Date in numeric format
##'
##' @seealso dateconvrt
##'
##' @export

datestrip <- function(sdate,format="%d-%b-%y %H:%M:%OS")
{
  date.strip <- unclass(as.POSIXct(strptime(sdate, format = format
      )))[1:length(sdate)]
  return(date.strip)
}
```

Listing A.13: `ssldtutils::exceldateconvrt` function

```
1
2  ##' Function to convert numeric Excel date values to string date
       values
3  ##'
4  ##' Utility function to convert numeric Excel date values to a
       string date
5  ##'
6  ##' @param idate : Numeric date from Excel.
7  ##' @param format : Format for string date (see ?strptime)
8  ##'
9  ##' @return Date in string format
10 ##'
11 ##' @seealso datestrip, dateconvrt
12 ##'
13 ##' @export
14
15
16 exceldateconvrt <- function(idate,format="%d-%b-%y %H:%M:%OS1"){
17
18   excel.date.convrt <- dateconvrt((idate*24*60*60)-50*3600,origin=
       "1900-01-01",format=format)
19   return(excel.date.convrt)
20 }
```

Listing A.14: `ssldtutils::posixdatestrip` function

```
1  ##' Convert POSIX  date values to numeric date values
2  ##'
3  ##' Utility function to convert POXIX date values to a numeric
       date (in the POSiX format)
4  ##'
5  ##' @param psxtime : Date in POXIX format.
6  ##' @param origin : Origin of date
7  ##'
8  ##' @return Date in numeric format
9  ##'
10 ##' @seealso dateconvrt, datestrip
11 ##'
12 ##' @export
13
14
15 posixdatestrip <- function(psxtime,origin = "1970-01-01") {
16   psxtime <-  unclass(as.POSIXct((psxtime), origin = "1970-01-01")
       )[1:length(psxtime)]
17   return(psxtime)
18 }
```

Listing A.15: `ssldtutils::getcurrtime` function

```
##' Function to return current date and time
##'
##' Utility function to return the current date and time in string
    format
##'
##' @param format : Format for string date (see ?strptime)
##'
##' @return Current datetime in string format
##'
##' @export

getcurrtime <- function(format= "%d-%b-%y %H:%M:%OS1")
{
  get.curr.time <- dateconvrt(Sys.time(),format=format)
  return(get.curr.time)

}
```

Listing A.16: `ssldtutils::comparedates` function

```
##' Function to compare two string dates
##'
##' Utility function to compare two dates in string format
##'
##' @param date1 : Date in string format
##' @param date2 : Date in string format
##' @param format : Format for string date (see ?strptime)
##'
##' @return 1 if date1 > date2, 0 if date1==date2, -1 if date1 <
    date2
##'
##' @export

comparedates <- function(date1,date2,format="%d-%b-%y %H:%M:%OS")
{
  date1num <- datestrip(date1,format)
  date2num <- datestrip(date2,format)
  if(date1num > date2num) {
    return(1)
  } else {
    if(date1num == date2num){
      return(0)
    } else {
      return(-1)
    }
  }
}
```

Listing A.17: `ssldtutils::daysbetween` function

```
##' Function to calculate number of days between two dates
##'
##' Utility function that returns the number of days between two
    dates in string format
##'
##' @param bdate : Start date in string format
##' @param edate : End date in string format
##' @param format : Format for string date (see ?strptime)
##'
##' @return Days between bdate and edate as numeric value
##'
##' @export

daysbetween <- function(bdate,edate, format="%d-%b-%y %H:%M:%OS")
{
    days.between <- floor(((datestrip(edate,format)-datestrip(
        bdate,format))/ 86400))
    return(days.between)
}
```

Listing A.18: `ssldtutils::nextday` function

```

##' Function that returns next day
##'
##' Utility function to return next day given a day. Usually used
    for stepping through days.
##'
##' @param sdate : Date in string format.
##' @param format : Format for string date (see ?strptime)
##'
##' @return Next day in string format
##'
##' @seealso prevday
##'
##' @export

nextday <- function(sdate,format="%d-%b-%y %H:%M:%OS")
{
  next.day <- dateconvrt(datestrip(sdate,format)+86400)
  return(next.day)
}
```

Listing A.19: `ssldtutils::prevday` function

```
1
2  ##' Function that returns previous day
3  ##'
4  ##' Utility function to return previous day given a day. Usually
       used for stepping through days.
5  ##'
6  ##' @param sdate : Date in string format.
7  ##' @param format : Format for string date (see ?strptime)
8  ##'
9  ##' @return Previous day in string format
10 ##'
11 ##' @seealso nextday
12 ##'
13 ##' @export
14
15 prevday <- function(sdate,format="%d-%b-%y %H:%M:%OS")
16 {
17   prev.day <- dateconvrt(datestrip(sdate,format)-86400)
18   return(prev.day)
19 }
```

Listing A.20: `ssldtutils::roundday` function

```
1
2  ##' Function that returns rounded day
3  ##'
4  ##' Utility function to return rounded day given a day. Usually
       used for stepping through days in conjunction with
5  ##' \code{nextday} or \code{prevday}.
6  ##'
7  ##' @param sdate : Date in string format.
8  ##' @param format : Format for string date (see ?strptime)
9  ##'
10 ##' @return Rounded day in string format
11 ##'
12 ##' @seealso ,roundmin,roundnmin,roundhour
13 ##'
14 ##' @export
15
16 roundday <- function(sdate,format="%d-%b-%y %H:%M:%OS")
17 {
18     round.day <- dateconvrt(datestrip(sdate,format)-(datestrip(
          sdate,format)-79200)%%86400)
19     return(round.day)
20 }
```

Listing A.21: `ssldtutils::roundhour` function

```
1
2  ##' Function that returns rounded datetime rounded (down) to
       nearest hour
3  ##'
4  ##' Utility function to return rounded (down) datetime to nearest
       hour.
5  ##'
6  ##' @param sdate : Date in string format.
7  ##' @param format : Format for string date (see ?strptime)
8  ##'
9  ##' @return Rounded datetime to hour in string format
10 ##'
11 ##' @seealso roundday,roundmin,roundnmin
12 ##'
13 ##' @export
14
15 roundhour <- function(sdate,format="%d-%b-%y %H:%M:%OS")
16 {
17     round.hour <- dateconvrt(datestrip(sdate,format)-(datestrip(
           sdate,format)-79200)%%3600,format="%d-%b-%y %H:%M:%S")
18     return(round.hour)
19 }
```

Listing A.22: `ssldtutils::roundnmin` function

```
1
2  ##' Function that returns rounded datetime rounded (down) to
       nearest n minute.
3  ##'
4  ##' Utility function to return rounded (down) datetime to nearest
       n minute.
5  ##'
6  ##' @param sdate : Date in string format.
7  ##' @param n : Numeric value for number of minutes to round down
       to (for example n=30 for halfhour)
8  ##' @param format : Format for string date (see ?strptime)
9  ##'
10 ##' @return Rounded datetime to nearest n minute in string format
11 ##'
12 ##' @seealso roundday, roundhour, roundmin
13 ##'
14 ##' @export
15
16
17 roundnmin <- function(sdate,n=1,format="%d-%b-%y %H:%M:%OS")
18 {
19     round.min <- dateconvrt(datestrip(sdate,format)-(datestrip(
           sdate,format)-79200)%%(60*n),format="%d-%b-%y %H:%M:%S")
20     return(round.min)
21 }
```

A.3 Database Interface Functions (ssldbutils)

Listing A.23: ssldbutils::dbconn function

```
#' Function that connects to database
#'
#' This function connects to an existing database.
#'
#' @param dbname ODBC Name
#' @param dbtype Database Type
#'
#' @return a connection object of class \code{"RODBC"} or \code{NULL} if connection not successful
#'
#' @export
#'
#
dbconn <- function(dbname,dbtype)
{

  if(dbtype=="Access") {
    ad <- odbcConnectAccess(dbname)
  } else {
    if(dbtype=="MySQL"){
      ad <- odbcConnect(dsn=dbname)

    } else {
      ad <- odbcDriverConnect(connection=paste("Driver=SQLite3 ODBC Driver;Database=",dbname,sep=""))
      sqlQuery(ad,"PRAGMA synchronous=OFF;")
    }
  }
  if(ad==-1){
    return(NULL)
  }
  return(ad)
}
```

Listing A.24: `ssldbutils::exptodb` function

```
#' Function that exports data to a table in the database
#'
#' Function that exports data to a table in the database
#'
#' @param dcsdat Data retrieved from DCS system
#' @param dbtable Table to export data to
#' @param dbname Name of database (ODBC) to export to
#' @param dbtype Type of database to export to (MySQL)
#' @param replace If \code{TRUE} replace INSERT with REPLACE in the
#' SQL statement
#'
#' @export


exptodb <-
  function(dcsdat,dbtable,dbname,dbtype,replace=FALSE)
  {
    ad <- dbconn(dbname,dbtype)
    if(!is.null(ad)){
      if(dbtype=="sqlite") {
        sqlQuery(ad,"BEGIN;")

        cnms <- "("
        for(i in colnames(dcsdat)[-ncol(dcsdat)])
          cnms <- paste(cnms,i,",",sep="")
        cnms <- paste(cnms,colnames(dcsdat)[ncol(dcsdat)],")",sep="")
        cnms <- paste("INSERT INTO ",dbtable,cnms,"VALUES (",sep=")
        if(length(colnames(dcsdat))>2){
          for(i in 1:nrow(dcsdat)) {
            sqlstr <- paste(cnms,"'",dcsdat[i,1],"'",sep="")
            for(j in 2 : (ncol(dcsdat)-1))
              sqlstr <- paste(sqlstr,",'",dcsdat[i,j],"'",sep="")

            sqlstr <- paste(sqlstr,",'",dcsdat[i,ncol(dcsdat)],"')",sep="")

            sqlQuery(ad,sqlstr)
          }
        } else {
          for(i in 1:nrow(dcsdat)) {
            sqlstr <- paste(cnms,"'",dcsdat[i,1],"'",sep="")
            sqlstr <- paste(sqlstr,",'",dcsdat[i,ncol(dcsdat)],
                            "')",sep="")
            sqlQuery(ad,sqlstr)
          }
        }
        sqlQuery(ad,"COMMIT;")
      } else {
```

```
48          cnms <- "(`"
49          for(i in colnames(dcsdat)[-ncol(dcsdat)])
50            cnms <- paste(cnms,i,"`,`",sep="")
51          cnms <- paste(cnms,colnames(dcsdat)[ncol(dcsdat)],"`)",sep
                ="")
52          if(!replace) {
53              cnms <- paste("INSERT INTO ",dbtable,cnms,"VALUES (",
                    sep=" ")
54          } else {
55              cnms <- paste("REPLACE INTO ",dbtable,cnms,"VALUES (",
                    sep=" ")
56          }
57          if(length(colnames(dcsdat))>2) {
58            for(i in 1:nrow(dcsdat)) {
59              sqlstr <- paste(cnms,"'",dcsdat[i,1],"'",sep="")
60              for(j in 2 : (ncol(dcsdat)-1))
61                sqlstr <- paste(sqlstr,",'",dcsdat[i,j],"'",sep="")
62
63              sqlstr <- paste(sqlstr,",'",
64                              dcsdat[i,ncol(dcsdat)],"')",sep="")
65              sqlQuery(ad,sqlstr)
66            }
67          } else {
68            for(i in 1:nrow(dcsdat)) {
69              sqlstr <- paste(cnms,"'",dcsdat[i,1],"'",sep="")
70              sqlstr <- paste(sqlstr,",'",dcsdat[i,ncol(dcsdat)],"')
                    ",
71                              sep="")
72              sqlQuery(ad,sqlstr)
73            }
74          }
75
76        }
77        odbcClose(ad)
78      } else {
79        return(NULL)
80      }
81    }
```

Listing A.25: `DBExp` C function

```
#ifndef DBUTIL_C
#define DBUTIL_C
#ifdef WIN64
#include <windows.h>
#else
#include <unistd.h>
#endif

#include <R.h>
#include <Rdefines.h>
#include <Rinternals.h>
#include <stdio.h>
#include <math.h>
#include <sql.h>
#include <sqltypes.h>
#include <sqlext.h>


SEXP DBExp(SEXP dbcon, SEXP TBName, SEXP ColVec, SEXP DatMat, SEXP
    nc, SEXP nr) {

    int i, j;
    const char *rconstr = CHAR(STRING_ELT(dbcon, 0));
    char sqlinit[5120] = {0};
    char sqlstmt[5120] = {0};
    SEXP SucRows;
    int *rsucrows;

    PROTECT(SucRows = allocVector(INTSXP, 1));
    rsucrows = INTEGER(SucRows);

    SQLHANDLE EnvHandle;
    SQLHANDLE ConHandle;
    SQLHANDLE StmtHandle;
    SQLRETURN rc;
    SQLCHAR ConString[255] = {0};
    SQLCHAR SQLStmt[5120] = {0};
    SQLCHAR outstring[1024] = {0};
#ifdef WIN32
    HWND desktopHandle = GetDesktopWindow();
#else
    HWND desktopHandle = NULL;
#endif
    strcpy((char *) ConString, rconstr);
    int *rnc = INTEGER(nc);
    int *rnr = INTEGER(nr);
    double *rdat = REAL(DatMat);

    sprintf(sqlinit, "INSERT INTO %s (", CHAR(STRING_ELT(TBName,
        0)));
```

```
49      sprintf(sqlinit, "%s %s", sqlinit, CHAR(STRING_ELT(ColVec, 0))
            );
50      for (i = 1; i < (*rnc); i++)
51          sprintf(sqlinit, "%s,%s", sqlinit, CHAR(STRING_ELT(ColVec,
                i)));
52
53      sprintf(sqlinit, "%s) VALUES (", sqlinit);
54      *rsucrows = 0;
55
56      rc = SQL_SUCCESS;
57      rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &
            EnvHandle);
58      //if(rc == SQL_SUCCESS)
59      // Set The ODBC Application Version To 3.x
60      if (rc == SQL_SUCCESS)
61          rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
62              (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);
63      // Allocate A Connection Handle
64      if (rc == SQL_SUCCESS)
65          rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle)
                ;
66
67      if (ConHandle != NULL) {
68          rc = SQLDriverConnect(ConHandle, desktopHandle, ConString,
                SQL_NTS, outstring, 1024, NULL, SQL_DRIVER_COMPLETE);
69          rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle,
70                  &StmtHandle);
71          sprintf(sqlstmt, "BEGIN;");
72          strcpy((char *) SQLStmt, sqlstmt);
73          rc = SQLExecDirect(StmtHandle, SQLStmt, SQL_NTS);
74          if (rc == SQL_SUCCESS)
75              fprintf(fdeb, "BEGIN \n");
76
77          for (i = 0; i < (*rnr); i++) {
78              sprintf(sqlstmt, "%s \'%-.16lf\'", sqlinit, rdat[i]);
79              for (j = 1; j < (*rnc); j++)
80                  sprintf(sqlstmt, "%s,\'%-.16lf\'", sqlstmt, rdat[i
                        + (*rnr) * j]);
81
82              sprintf(sqlstmt, "%s);", sqlstmt);
83
84              strcpy((char *) SQLStmt, sqlstmt);
85              // Prepare And Execute The SQL Statement
86              rc = SQLExecDirect(StmtHandle, SQLStmt, SQL_NTS);
87              *rsucrows += (rc == SQL_SUCCESS);
88
89          }
90          sprintf(sqlstmt, "COMMIT;");
91          strcpy((char *) SQLStmt, sqlstmt);
92          rc = SQLExecDirect(StmtHandle, SQLStmt, SQL_NTS);
93          if (StmtHandle != NULL)
94              SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
```

```
95          rc = SQLDisconnect(ConHandle);
96      }
97
98      if (ConHandle != NULL)
99          SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
100
101     if (EnvHandle != NULL)
102         SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
103
104
105     UNPROTECT(1);
106
107     return SucRows;
108
109 }
110
111
112 #endif          //  #ifndef DBUTIL_C
```

Listing A.26: `ssldbutils::exptodbc` function

```
1  #' Function that exports data to a table in the database
2  #'
3  #' Function that exports data to a table in the database. Uses a C
       dll for odbc interface,
4  #' which leads to large performance increase. Can however
       currently not
5  #' accept character colums in \code{dcsdat}.
6  #'
7  #' @param dcsdat Data retrieved from DCS system
8  #' @param dbtable Table to export data to
9  #' @param dbname Name of database to export to
10 #' @param dbtype Type of database to export to
11 #'
12 #' @export
13
14 exptodbc <-
15   function(dcsdat,dbtable,dbname,dbtype) {
16     if(dbtype == "sqlite") {
17       constr <- paste("DSN=SQLite3 Datasource;Database=",dbname,
             sep="")
18     } else {
19       constr <- paste("DSN=",dbname,sep="")
20
21     }
22     print(constr)
23     return(.Call("DBExp",constr,dbtable,colnames(dcsdat),
24                  as.matrix(dcsdat),as.integer(ncol(dcsdat)),
25                  as.integer(nrow(dcsdat))))
26   }
```

Listing A.27: `ssldbutils::errlog` function

```
#' Function to log error messages to log table in database
#'
#' This function logs an error message to a log table in the database.
#'
#' @param dbname Database to log the error to.
#' @param dbtype Type of database.
#' @param errtable Name of error log table.
#' @param appname Name of R package that encountered an error.
#' @param fnname Name of R function that encounterd an error.
#' @param errmsg Error message to log
#'
#'
#' @export
#'
#

errlog <- function(dbname,dbtype,errtable,appname,fnname,errmsg){
  ad <- dbconn(dbname,dbtype)
  if(!is.null(ad)){
    sqlstr <- paste("INSERT INTO `",errtable,
                    "` (`DateTime`,`AppName`,`FuncName`,`ErrMsg`)
                       VALUES ('",
                    datestrip(getcurrtime()),"', '",appname,"', '",
                       ',
                    fnname,"', '",errmsg, "');",sep="")
    res <- sqlQuery(ad,sqlstr)

    close(ad)
    return(res)
  } else {
    write.table(t(c(datestrip(getcurrtime()),appname,fnname,errmsg
        )),paste(errtable,".csv",sep=""),
                append=TRUE,sep=",",row.names=FALSE,col.names=
                   FALSE)
    return(NULL)
  }

}
```

Listing A.28: `ssldbutils::getdbtables` function

```
##' Function to retrieve database tables between two dates
##'
##' @param dbtables : Tables to retrieve
##' @param bdate : Start date for retrieval
##' @param edate : End date for retreeval
##' @param dbname : Database name to retrieve from
##' @param dbtype : Database type to retrieve from
##' @param datetime : Column which contains the time values
##'
##' @export
##



getdbtables <- function(dbtables, bdate, edate,dbname,dbtype,
    datetime) {
     ntb <- length(dbtables)
     retdbtables <- list("Success"=FALSE)
     if(ntb > 0) {

          for(i in dbtables) {
               print(i)

               tbname <- i
               if(comparedates(bdate,getmaxtime(tbname,dbname,dbtype,
                   datetime))==1) {
                    retdbtables$Message = paste(bdate, " greater than
                        maximim date in ",tbname, " (",
                         getmaxtime(tbname,dbname,dbtype,datetime),")",
                             sep="")

                    return(retdbtables)

               } else {

                    if(comparedates(getmintime(tbname,dbname,dbtype,
                        datetime),edate)==1) {
                         retdbtables$Message = paste(edate, " smaller
                             than minimum date in ",tbname, " (",
                              getmintime(tbname,dbname,dbtype,datetime),
                                  ")",sep="")

                         return(retdbtables)


                    }
                    if(comparedates(edate,getmaxtime(tbname,dbname,
                        dbtype,datetime))==1) {
                         retdbtables$Message = paste(edate, " greater
                             than maximim date in ",tbname, " (",
```

```
42                          getmaxtime(tbname,dbname,dbtype,datetime),
                                ")",sep="")
43
44                      return(retdbtables)
45
46                  }
47
48                  ad <- dbconn(dbname,dbtype)
49                  tb <- sqlQuery(ad,paste("SELECT * FROM ",tbname,"
                        WHERE ",datetime," BETWEEN ",
50                                          datestrip(bdate), " AND ",
                                                datestrip(edate),"
                                               ORDER BY ",
51                                          datetime,";",sep=""))
52
53                  if(is.null(nrow(tb))){
54                      retdbtables$Message = paste("No Values
                            Returned For Table ",tbname,sep="")
55                      close(ad)
56                      return(retdbtables)
57                  }
58                  retdbtables[[tbname]] <- tb
59                  rm(tb)
60                  close(ad)
61              }
62          }
63
64      } else {
65          retdbtables$Message = "No Tables Selected"
66          return(retdbtables)
67      }
68      retdbtables$Success=TRUE
69      retdbtables$Message = "Data Succesfully Retrieved"
70
71      return(retdbtables)
72  }
```

Listing A.29: `ssldbutils::droptables` function

```
#' Function to drop tables from database
#'
#' This function is used to delete tables from the database. It can run in test mode, which
#' will print the query to the console, or in production mode which will run the code and
#' delete the tables.
#'
#' @param TableNames Table to delete from the database.
#' @param testonly Choose to run in test mode, or if FALSE production mode.
#' @export
#'
droptables <- function(TableNames,dbname,dbtype,testonly=FALSE){
  ad <- dbconn(dbname,dbtype)
  for(i in TableNames){
    if(testonly) {
      print(paste("DROP TABLE ",i,";",sep=""))
    } else {
      sqlQuery(ad,paste("DROP TABLE ",i,";",sep=""))
    }
  }
  close(ad)
}
```

Listing A.30: `ssldbutils::tblexist` function

```
#' Function to check if table exist in the database
#'
#' This function tests if a table exist in a database.
#'
#' @param dbname Database Name
#' @param dbtype Database Type
#' @param table Table Name
#' @param schemaname Schema in database where table should exist.
#'
#' @return a 1 if table exist, else a 0. \code{NULL} if error.
#' @export
#'
#

tblexist <- function(dbname,dbtype,table,schemaname=dbname){
  ad <- dbconn(dbname,dbtype)
  if(is.null(ad)) {
    cat("Could Not Connect to Database : ",dbname,
        " in function tblexist \n",sep="")
    return(ad)

  }
  sqlstr <- paste("SELECT COUNT(*)
                  FROM information_schema.tables
```

```
                        WHERE table_schema = '",schemaname,"'
                        AND table_name = '",table,"';",sep="")
  ##print(sqlstr)
  res <- sqlQuery(ad,sqlstr)
  return(as.numeric(res))
}
```

Listing A.31: `ssldbutils::tblrows` function

```
#' Function to return the number of rows in a table
#'
#' This function returns the number of rows in a table.
#'
#' @param dbname Database Name
#' @param dbtype Database Type
#' @param table Table Name
#' @param column Column to do the count on
#' @param schemaname Schema in database
#' @return Number of rows in the Table or NULL if error.
#' @export
#'
#

tblrows <- function(dbname,dbtype,table,column,schemaname){
  ad <- dbconn(dbname,dbtype)
  if(is.null(ad)) {
    cat("Could Not Connect to Database : ",dbname,
        " in function tblrows \n",sep="")
    return(ad)

  }
  if(tblexist(dbname,dbtype,table,schemaname)) {
    sqlstr <- paste("SELECT COUNT(`",column,"`)
                    FROM `",table,"`;",sep="")
    res <- sqlQuery(ad,sqlstr)
    close(ad)
    return(as.numeric(res))
  } else {
    cat("Table : ",table," does not exist in database : ",
        dbname," in function tblrows \n",sep="")
    close(ad)
    return(NULL)
  }
}
```

Listing A.32: `ssldbutils::getallmax` function

```
##' Function to get maximum datetimes for list of tables
##'
##'
##' Function to retrieve maximum datetime for list of tables.
##'
##' @param tbls : vector of table names.
##' @param dbname : Schema name in database
##' @param dbtype : Type of database
##' @param datetime : Column name of datetime values.
##'
##' @return Vector of maximum values for the tables.
##'
##' @seealso getmaxtime, getmintime
##'
##' @export
##'

getallmax <- function(tbls,dbname,dbtype,datetime){

  tbmax <- vector("numeric",length(tbls))
  for(i in 1:length(tbls)){
    tbmax[i] <- datestrip(getmaxtime(tbls[i],dbname,dbtype,
        datetime))

  }
  return(dateconvrt(tbmax))

}
```

Listing A.33: `ssldbutils::getmaxmax` function

```
##' Funtion to retrieve maximum of all the maximum datetimes for
    list of tables
##'
##'
##' Function to retrieve maximum of all the maximums for list of
    tables.
##'
##' @param tbls : vector of table names.
##' @param dbname : Schema name in database
##' @param dbtype : Type of database
##' @param datetime : Column name of datetime values.
##'
##' @return Index in tbls that has maximum datetime.
##'
##' @seealso getmaxtime, getmintime
##'
##' @export
##'

getmaxmax <- function(tbls,dbname,dbtype,datetime){
```

```
19
20    tbmax <- vector("numeric",length(tbls))
21    for(i in 1:length(tbls)){
22      tbmax[i] <- datestrip(getmaxtime(tbls[i],dbname,dbtype,
          datetime))
23
24    }
25    return(which.max(tbmax))
26
27  }
```

Listing A.34: `ssldbutils::getminmax` function

```
1  ##' Funtion to retrieve the minimum of all the maximum datetimes
       for list of tables
2  ##'
3  ##'
4  ##' Function to retrieve minimum of all the maximums for list of
       tables.
5  ##'
6  ##' @param tbls : vector of table names.
7  ##' @param dbname : Schema name in database
8  ##' @param dbtype : Type of database
9  ##' @param datetime : Column name of datetime values.
10 ##'
11 ##' @return Index in tbls that has minimum maximum datetime.
12 ##'
13 ##' @seealso getmaxtime, getmintime
14 ##'
15 ##' @export
16 ##'
17
18 getminmax <- function(tbls,dbname,dbtype,datetime){
19   tbmax <- vector("numeric",length(tbls))
20   for(i in 1:length(tbls)){
21     tbmax[i] <- datestrip(getmaxtime(tbls[i],dbname,dbtype,
         datetime))
22
23   }
24   return(which.min(tbmax))
25
26 }
```

Listing A.35: `ssldbutils::getmaxtime` function

```
##' Function to retrieve the max time in table
##'
##' Function to retrieve maximum time in table. Normally used to
    retrieve last value in table from database.
##'
##' @param dbtable : Table in database that maximum time is needed
     for.
##' @param dbname : Schema name in database
##' @param dbtype : Type of database
##' @param datetime : Column name of datetime values.
##'
##' @seealso getmintime
##'
##' @export

getmaxtime <- function(dbtable,dbname,dbtype,datetime)
{
    ad <- dbconn(dbname,dbtype)
    sqlmax <- paste("SELECT MAX(",datetime,") from ",dbtable,sep="
        ")
    max.time <- sqlQuery(ad,sqlmax)
    getMaxTime <- dateconvrt(as.double(max.time))
    close(ad)
    return(getMaxTime)
}
```

Listing A.36: `ssldbutils::getmintime` function

```
##' Function to retrieve the min time in table
##'
##' Function to retrieve minimum time in table. Normally used to
    retrieve first value in table from database.
##'
##' @param dbtable : Table in database that minimum time is needed
     for.
##' @param dbname : Schema name in database
##' @param dbtype : Type of database
##' @param datetime : Column name of datetime values.
##'
##' @seealso getmaxtime
##'
##' @export
##'
getmintime <- function (dbtable,dbname,dbtype,datetime)
{
    ad <- dbconn(dbname,dbtype)
    sqlmin <- paste("SELECT MIN(", datetime, ") from ", dbtable,
        sep = "")
    min.time <- sqlQuery(ad, sqlmin)
    getMinTime <- dateconvrt(as.double(min.time))
    close(ad)
```

```
22      return(getMinTime)
23  }
```

A.4 Local DCS Data Interface Functions (ssldcsutils)

Listing A.37: ssldcsutils::retrdscdata function

```
#' Function to retrieve DCS Data from Database
#'
#'Retrieve Data from database between bdate and edate for tags.
#'for \code{ssize=NULL} only one tag can be retrieved
#'
#' @param bdate : Start data for download
#' @param edate : End date for download
#' @param tags : Tags to download
#' @param ssize : Step size for aggregation in minutes, if \code{NULL} return raw data
#' @param stats : Statistics for data aggregation
#' @param dbname : Name of database to retrieve data from
#' @param dbtype : Type of database
#' @param datetime : DateTime column in database
#'
#'
#' @export



retrdcsdata <-
  function(bdate,edate,tags,ssize=NULL,stats=0,dbname,dbtype="MySQL",datetime="DateTime")
  {
    options(stringsAsFactors=FALSE)
    ##strip all sting delimiters from tags
    tags <- gsub("'","",tags)
    tags <- gsub("`","",tags)
    ##Add ` for table names.
    tags <- paste("`",tags,"`",sep="")
    tags <- unique(tags)
    ad <- dbconn(dbname,dbtype)
    tb <- list()
    for(i in tags){
      if(is.null(ssize)) {
        bd <- bdate
        ed <- edate
      } else {
        maxd <- getmaxtime(i,dbname,dbtype,datetime)
        mind <- getmintime(i,dbname,dbtype,datetime)

        if(comparedates(edate,maxd)<0){ ## edate < max date. update to value above maxdate
          sqstr <- paste("SELECT MIN(DateTime) FROM ",i,"WHERE DateTime > ",datestrip(edate),";")
          ed <- dateconvrt(as.numeric(sqlQuery(ad,sqstr)))
        } else {
```

```
44            ed <- edate
45            if(comparedates(edate,maxd)>0){
46              message(paste(edate," bigger than maximum date in
                    database ",maxd," for tag ",i,"!",sep=""))
47            }
48          }
49
50          if(comparedates(bdate,mind)<0){ ## edate < max date.
                update to value above maxdate
51            bd <- bdate
52            message(paste(bdate," smaller than minimum date in
                  database ",mind," for tag ",i,"!",sep=""))
53
54          } else {
55            if(comparedates(bdate,mind)==0){
56              bd <- bdate
57            } else {
58              if(comparedates(bdate,maxd) == 0){
59                bd  <- maxd
60              } else {
61                sqstr <- paste("SELECT MAX(DateTime) FROM ",i,"WHERE
                      DateTime < ",datestrip(bdate),";")
62                bd <- dateconvrt(as.numeric(sqlQuery(ad,sqstr)))
63              }
64            }
65          }
66
67
68        }
69        if(is.null(ssize)) {
70          sqstr <- paste("SELECT * FROM ",i," WHERE ",datetime,"
                BETWEEN '",
71                         datestrip(bd),"' AND '",
72                         datestrip(ed),"' ORDER BY ",datetime,";",
                             sep="")
73        } else {
74          sqstr <- paste("SELECT * FROM ",i," WHERE ",datetime,"
                BETWEEN '",
75                         datestrip(bd)-1,"' AND '",
76                         datestrip(ed)+1,"' ORDER BY ",datetime,";",
                             sep="")
77
78        }
79        tb[[i]] <- sqlQuery(ad,sqstr)
80      }
81      odbcClose(ad)
82
83      if(is.null(ssize)) {
84        r <- 1
85        retval <- NULL
86        for(i in names(tb)){
87          if(r == 1){
```

```
88            retval <- tb[[i]][,1:2]
89            retval[,2] <- retval[,2] * (!tb[[i]][,3])
90            r <- r + 1
91          } else {
92            retval <- cbind(retval,tb[[i]][,2]*(!tb[[i]][,3]))
93            r <- r + 1
94          }
95
96
97        }
98        colnames(retval) <- c(datetime,tags)
99      } else {
100       r <- 1
101       retval <- NULL
102
103       for(i in names(tb)) {
104
105         tmw <- twstats(as.matrix(tb[[i]]),bdate,edate,stats,ssize*
              60)
106         if(r==1){
107           retval <- tmw
108           retval[,1] <- datestrip(roundmin(dateconvrt(retval[,1]))
                )
109           r <- r + 1
110         } else {
111           retval <- cbind(retval,tmw[,2])
112
113         }
114
115       }
116       colnames(retval) <- c(datetime,tags)
117     }
118     return(retval)
119   }
```

Listing A.38: `TimeWeigtedStats` C function

```
1
2  #ifndef DCSUTIL_C
3  #define DCSUTIL_C
4
5  #include <R.h>
6  #include <Rdefines.h>
7  #include <Rinternals.h>
8  #include <stdio.h>
9  #include <math.h>
10
11
12 double lininterpol(double x1, double x2, double x3, double y1,
       double y3);
13 double AVE(double *times, double *vals, double *stat, int nvals,
       double stime, double etime);
14 double VAR(double *times, double *vals, double *stat, int nvals,
       double stime, double etime);
15 double SD(double *times, double *vals, double *stat, int nvals,
       double stime, double etime);
16 double FVAL(double *times, double *vals, double *stat, int nvals,
       double stime, double etime);
17 double MIN(double *times, double *vals, double *stat, int nvals,
       double stime, double etime);
18 double MAX(double *times, double *vals, double *stat, int nvals,
       double stime, double etime);
19
20
21 /*
22   Ruan Rossouw
23   23/08/2011
24   Calculates Time Weighted statistics from a dataset
25  * It is important to make sure that rawdat[0,0] <= stime and
26  * rawdata[nrow-1,0] >= etime
27  * A check is added in the code, and the rawdat[0,0] will be
       assigned stime and
28  * rawdata[nrow-1,0] = etime if not the case. This should however
       not happen.
29   rawdat : Raw data as downloaded form database system. First
       column
30   is time in the R time format i.e. seconds from a certain date.
       Data
31   sorted in increasing order according to time,third column is
       status of data.
32   nrow : The number of rows in rawdat
33   ncol : The number of cols in rawdat
34   stepsize : Size of the steps for interpolation.
35   starttime : Start time for interpolation.
36   endtime : End time for interpolation.
37   stat : What statistic to use 0 for AVG, 1 for VAR, 2 for SD, 3
       for MIN, 4 for MAX and 10 for FVAL
38  * This will return -9999 if no "good" data is available in range.
```

```
39 This is needed for certain "shortcomings" in PI interface....
40  */
41
42 SEXP TimeWeightStats(SEXP rawdat, SEXP nrow, SEXP ncol, SEXP
       stepsize, SEXP starttime, SEXP endtime, SEXP stat) {
43
44
45     int i, k, r, *rnrow = INTEGER(nrow), *rncol = INTEGER(ncol),
           tlength,
46             *rstat = INTEGER(stat), nval, stii, etii, rst;
47     double *rrawdat = REAL(rawdat), *rcombdat, *rstime = REAL(
           starttime), *retime = REAL(endtime), *rstsize = REAL(
           stepsize), sti, eti;
48
49     //function pointer to assign to the appropriate statistic from
            stat
50     //double (*statf)(double *times, double *vals, int nvals,
           double stime, double etime);
51     double (*statfarr[1])(double *times, double *vals, double *
           stat, int nvals, double stime, double etime);
52
53     SEXP CombDat;
54
55
56     tlength = ((*retime - *rstime) / *rstsize);
57
58     PROTECT(CombDat = allocMatrix(REALSXP, tlength, 2));
59     rcombdat = REAL(CombDat);
60
61     for (k = 1; k < 2; k++) {
62         rst = rstat[k - 1];
63         switch (rst) {
64             case 0: statfarr[k - 1] = AVE;
65                 break;
66             case 1: statfarr[k - 1] = VAR;
67                 break;
68             case 2: statfarr[k - 1] = SD;
69                 break;
70             case 3: statfarr[k - 1] = MAX;
71                 break;
72             case 4: statfarr[k - 1] = MIN;
73                 break;
74             case 10: statfarr[k - 1] = FVAL;
75                 break;
76             default: Rprintf("Unimplemented Statistic Selected \n"
                   );
77                 exit(-1);
78                 break;
79
80         }
81     }
82
```

```
83
84
85      for (i = 0; i < (tlength); i++)
86          rcombdat[i] = 0;
87
88
89      //These if statements should never evaluate to TRUE
90      if (rrawdat[0] > *rstime) {
91          rrawdat[0] = *rstime;
92          Rprintf("First value in rawdat > stime! Updated to stime \
                n");
93      }
94      if (rrawdat[*rnrow - 1] < *retime) {
95          rrawdat[*rnrow - 1] = *retime;
96          Rprintf("Last value in rawdat < etime! Updated to etime \n
                ");
97      }
98
99
100     //r: index into the CombDat matrix.
101
102     r = 0;
103     stii = 0;
104     etii = 0;
105     for (r = 0; r < tlength; r++) {
106
107         rcombdat[r] = *rstime + r * (*rstsize);
108         sti = rcombdat[r];
109         eti = sti + (*rstsize);
110         while (rrawdat[stii] < sti)
111             stii++;
112         stii--;
113         while (rrawdat[etii] < eti)
114             etii++;
115         nval = etii - stii + 1;
116         for (k = 1; k < 2; k++)
117             rcombdat[r + tlength * k] = (*statfarr[k - 1])(&
                    rrawdat[stii], &rrawdat[stii + (*rnrow) * k], &
                    rrawdat[stii + (*rnrow) * (k + 1)], nval, sti, eti)
                    ;
118
119
120     }
121
122
123     UNPROTECT(1);
124
125     return (CombDat);
126 }
127
128 /*
```

```
129  * Calculates the time weighted average of vals by the integral
         aggregate
130  * calculation method
131  * The times[0] value must be <= stime, and time[nvals-1] must be
         >= etime
132  * This is a property of the calculations
133  */
134 double AVE(double *times, double *vals, double *stat, int nvals,
        double stime, double etime) {
135     double sval, eval, sti, eti, sum, sstat, estat, count;
136     int i;
137     if(stat[1]==0){
138     sval = lininterpol(times[0], stime, times[1], vals[0], vals
            [1]);
139     } else {
140         sval = vals[0];
141     }
142     sstat = stat[0];
143     //check if any actual values in range, else interpolate eval
            as well.
144     if (nvals > 2) {
145         eval = vals[1];
146         estat = stat[1];
147         sti = stime;
148         eti = times[1];
149     } else {
150         eval = lininterpol(times[0], etime, times[1], vals[0],
                vals[1]);
151         estat = stat[1];
152         sti = stime;
153         eti = etime;
154     }
155
156     sum = 0;
157     count = 0;
158     if (sstat == 0) {
159
160         if (estat == 0) {
161             sum = (sval + eval)*(eti - sti) / 2;
162         } else {
163             sum = (sval + sval)*(eti - sti) / 2;
164         }
165         count += eti - sti;
166     }
167     //if nvals > 2 there is actual values in range.
168     if (nvals > 2) {
169         for (i = 1; i < (nvals - 1); i++) {
170             sti = eti;
171             sval = eval;
172             sstat = estat;
173             eti = times[i];
174             eval = vals[i];
```

```
175             estat = stat[i];
176         if (sstat == 0) {
177                 if (estat == 0) {
178                     sum += (sval + eval)*(eti - sti) / 2;
179                 } else {
180                     sum += (sval + sval)*(eti - sti) / 2;
181                 }
182                 count += eti - sti;
183             }
184 
185         }
186         sti = eti;
187         sval = eval;
188         sstat = estat;
189         estat = stat[nvals - 1];
190         eval = lininterpol(sti, etime, times[nvals - 1], sval,
                vals[nvals - 1]);
191         eti = etime;
192 
193         if (sstat == 0) {
194             if (estat == 0) {
195                 sum += (sval + eval)*(eti - sti) / 2;
196             } else {
197                 sum += (sval + sval)*(eti - sti) / 2;
198             }
199             count += eti - sti;
200         }
201 
202     }
203     if (count > 0) {
204         return sum / count;
205     } else {
206         return -9999;
207     }
208 }
209 
210 /*
211  * Calculates the time weighted variance of vals by the integral
       aggregate
212  * calculation method
213  * The times[0] value must be <= stime, and time[nvals-1] must be
       >= etime
214  * This is a property of the calculations
215  */
216 double VAR(double *times, double *vals, double *stat, int nvals,
       double stime, double etime) {
217     double sval, eval, sti, eti, sum, vsum, count, sstat, estat;
218     int i;
219     if(stat[1]==0){
220     sval = lininterpol(times[0], stime, times[1], vals[0], vals
           [1]);
221     } else {
```

```
222             sval = vals[0];
223         }
224         sstat = stat[0];
225         //check if any actual values in range, else interpolate eval
                as well.
226         if (nvals > 2) {
227             eval = vals[1];
228             estat = stat[1];
229             sti = stime;
230             eti = times[1];
231         } else {
232             eval = lininterpol(times[0], etime, times[1], vals[0],
                    vals[1]);
233             estat = stat[1];
234             sti = stime;
235             eti = etime;
236         }
237
238         sum = 0;
239         vsum = 0;
240         count = 0;
241         if (sstat == 0) {
242             if (estat == 0) {
243                 sum = (sval + eval)*(eti - sti) / 2;
244                 vsum = (eti - sti)*((((eval - sval)*(eval - sval)) /
                        3)+(eval - sval) * sval + sval * sval);
245                 count += eti - sti;
246             } else {
247                 sum = (sval + sval)*(eti - sti) / 2;
248                 vsum = (eti - sti)*((((sval - sval)*(sval - sval)) /
                        3)+(sval - sval) * sval + sval * sval);
249                 count += eti - sti;
250             }
251
252         }
253         //if nvals > 2 there is actual values in range.
254         if (nvals > 2) {
255             for (i = 1; i < (nvals - 1); i++) {
256                 sti = eti;
257                 sval = eval;
258                 sstat = estat;
259                 eti = times[i];
260                 eval = vals[i];
261                 estat = stat[i];
262                 if (sstat == 0) {
263                     if (estat == 0) {
264                         sum += (sval + eval)*(eti - sti) / 2;
265                         vsum += (eti - sti)*((((eval - sval)*(eval -
                                sval)) / 3)+(eval - sval) * sval + sval *
                                sval);
266                         count += eti - sti;
267                     } else {
```

```
268                     sum += (sval + sval)*(eti - sti) / 2;
269                     vsum += (eti - sti)*((((sval - sval)*(sval -
                            sval)) / 3)+(sval - sval) * sval + sval *
                            sval);
270                     count += eti - sti;
271                 }
272 
273             }
274 
275         }
276         sti = eti;
277         sval = eval;
278         sstat = estat;
279         estat = stat[nvals - 1];
280         eval = lininterpol(sti, etime, times[nvals - 1], sval,
                vals[nvals - 1]);
281         eti = etime;
282 
283         if (sstat == 0) {
284             if (estat == 0) {
285                 sum += (sval + eval)*(eti - sti) / 2;
286                 vsum += (eti - sti)*((((eval - sval)*(eval - sval)
                        ) / 3)+(eval - sval) * sval + sval * sval);
287                 count += eti - sti;
288             } else {
289                     sum += (sval + sval)*(eti - sti) / 2;
290                     vsum += (eti - sti)*((((sval - sval)*(sval -
                            sval)) / 3)+(sval - sval) * sval + sval *
                            sval);
291                     count += eti - sti;
292                 }
293         }
294 
295 
296     }
297     if (count > 0) {
298         return (vsum - ((sum * sum) / count)) / count;
299     } else {
300         return -9999;
301     }
302 }
303 
304 /*
305  * Calculates the time weighted standard deviation of vals by the
        integral aggregate
306  * calculation method
307  * The times[0] value must be <= stime, and time[nvals-1] must be
        >= etime
308  * This is a property of the calculations
309  */
310 double SD(double *times, double *vals, double *stat, int nvals,
    double stime, double etime) {
```

```
311     double var;
312     var = VAR(times, vals, stat, nvals, stime, etime);
313     if(var>0){
314         return sqrt(var);
315     } else {
316         return -9999;
317     }
318 
319 }
320 
321 
322 /*
323  * Returns the first value in the range or the first value before
        (Value on hold).
324  *
325  * The times[0] value must be <= stime, and time[nvals-1] must be
        >= etime
326  * This is a property of the calculations
327  */
328 double FVAL(double *times, double *vals, double *stat, int nvals,
        double stime, double etime) {
329     double fval;
330     int i;
331     int statflag = 0;
332     //Take first value in range, else take first value before
            range.
333     if (nvals > 2) {
334         for (i = 1; i < nvals; i++) {
335             if (stat[i] == 0) {
336                 fval = vals[i];
337                 statflag = 1;
338                 break;
339 
340             }
341         }
342         if (stat[0] == 0) {
343             fval = vals[0];
344             statflag = 1;
345 
346         }
347 
348     } else {
349         if (stat[0] == 0) {
350             fval = vals[0];
351             statflag = 1;
352         } else {
353             if (stat[1] == 0) {
354                 fval = vals[0];
355                 statflag = 1;
356 
357             }
358         }
```

```
359     }
360     if (statflag == 1) {
361         return fval;
362     } else {
363         return -9999;
364     }
365
366 }
367
368
369 /*
370  * Returns the max value in the range or the first value before.
371  * The times[0] value must be <= stime, and time[nvals-1] must be
        >= etime
372  * This is a property of the calculations
373  */
374
375 double MAX(double *times, double *vals, double *stat, int nvals,
        double stime, double etime) {
376     double sval, eval, sti, eti, max, sstat, estat, count;
377     int i;
378     if(stat[1]==0){
379     sval = lininterpol(times[0], stime, times[1], vals[0], vals
            [1]);
380     } else {
381         sval = vals[0];
382     }
383     sstat = stat[0];
384     //check if any actual values in range, else interpolate eval
            as well.
385     if (nvals > 2) {
386         eval = vals[1];
387         estat = stat[1];
388         sti = stime;
389         eti = times[1];
390     } else {
391         eval = lininterpol(times[0], etime, times[1], vals[0],
                vals[1]);
392         estat = stat[1];
393         sti = stime;
394         eti = etime;
395     }
396
397     max = -1e-9;
398     count = 0;
399     if (sstat == 0) {
400
401         if (estat == 0) {
402             if(eval > sval){
403             max = eval;
404             } else {
405                 max = sval;
```

```
406                 }
407             } else {
408                 max = sval;
409             }
410             count += eti - sti;
411         }
412         if (nvals > 2) {
413             for (i = 1; i < (nvals - 1); i++) {
414                 sti = eti;
415                 sval = eval;
416                 sstat = estat;
417                 eti = times[i];
418                 eval = vals[i];
419                 estat = stat[i];
420                 if (sstat == 0) {
421                     if (estat == 0) {
422                         if(eval > sval){
423                             if(max < eval){
424                                 max = eval;
425                             }
426                         } else {
427                             if(max < sval) {
428                                 max = sval;
429                             }
430                         }
431             } else {
432                         if(max < sval){
433                             max = sval;
434                         }
435             }
436                     count += eti - sti;
437                 }
438 
439             }
440             sti = eti;
441             sval = eval;
442             sstat = estat;
443             estat = stat[nvals - 1];
444             eval = lininterpol(sti, etime, times[nvals - 1], sval,
                    vals[nvals - 1]);
445             eti = etime;
446 
447             if (sstat == 0) {
448                 if (estat == 0) {
449                     if(eval > sval){
450                         if(max < eval){
451                             max = eval;
452                         }
453                     } else {
454                         if(max < sval) {
455                             max = sval;
456                         }
```

```
457                         }
458             } else {
459                 if(max < sval){
460                             max = sval;
461                     }
462             }
463             count += eti - sti;
464         }
465     }
466     if (count > 0) {
467         return max;
468     } else {
469         return -9999;
470     }
471 }
472
473
474
475 /*
476  * Returns the min value in the range or the first value before.
477  * calculation method
478  * The times[0] value must be <= stime, and time[nvals-1] must be
       >= etime
479  * This is a property of the calculations
480  */
481 double MIN(double *times, double *vals, double *stat, int nvals,
       double stime, double etime) {
482     double sval, eval, sti, eti, min, sstat, estat, count;
483     int i;
484     if(stat[1]==0){
485     sval = lininterpol(times[0], stime, times[1], vals[0], vals
           [1]);
486     } else {
487         sval = vals[0];
488     }
489     sstat = stat[0];
490     //check if any actual values in range, else interpolate eval
           as well.
491     if (nvals > 2) {
492         eval = vals[1];
493         estat = stat[1];
494         sti = stime;
495         eti = times[1];
496     } else {
497         eval = lininterpol(times[0], etime, times[1], vals[0],
               vals[1]);
498         estat = stat[1];
499         sti = stime;
500         eti = etime;
501     }
502
503     min = 1e9;
```

```
504     count = 0;
505     if (sstat == 0) {
506
507         if (estat == 0) {
508             if(eval < sval){
509             min = eval;
510             } else {
511                 min = sval;
512             }
513         } else {
514             min = sval;
515         }
516         count += eti - sti;
517     }
518     if (nvals > 2) {
519         for (i = 1; i < (nvals - 1); i++) {
520             sti = eti;
521             sval = eval;
522             sstat = estat;
523             eti = times[i];
524             eval = vals[i];
525             estat = stat[i];
526             if (sstat == 0) {
527                 if (estat == 0) {
528                     if(eval < sval){
529                         if(min > eval){
530                             min = eval;
531                         }
532                     } else {
533                         if(min > sval) {
534                             min = sval;
535                         }
536                     }
537         } else {
538                     if(min > sval){
539                         min = sval;
540                     }
541         }
542                 count += eti - sti;
543             }
544
545         }
546         sti = eti;
547         sval = eval;
548         sstat = estat;
549         estat = stat[nvals - 1];
550         eval = lininterpol(sti, etime, times[nvals - 1], sval,
                vals[nvals - 1]);
551         eti = etime;
552
553         if (sstat == 0) {
554             if (estat == 0) {
```

```
555                 if(eval < sval){
556                             if(min > eval){
557                                 min = eval;
558                             }
559                         } else {
560                                 if(min > sval) {
561                                     min = sval;
562                                 }
563                         }
564             } else {
565                 if(min > sval){
566                             min = sval;
567                     }
568             }
569             count += eti - sti;
570         }
571 
572     }
573     if (count > 0) {
574         return min;
575     } else {
576         return -9999;
577     }
578 }
579 
580 
581 
582 double lininterpol(double x1, double x2, double x3, double y1,
        double y3) {
583     double y2;
584     if (x3 == x1) {
585         return y3;
586     } else {
587         y2 = ((y3 - y1) / (x3 - x1))*(x2 - x1) + y1;
588     }
589     return y2;
590 }
591 
592 #endif          //  #ifndef DCSUTIL_C
```

Listing A.39: `ssldcsutils::twstats` function

```
#' Function to calculate Time Weighted Statistics for DCS Data
#'
#' Calculates Time Weighted Statistics for DCS Data
#'
#' @param dcsdat Data retrieved from DCS system
#' @param bdate Start date for downloaded data
#' @param edate End date for downloaded data
#' @param stat Statistic to perform
#' @param ssize Step size for returned data
#'
#' @export


twstats <- function(dcsdat,bdate,edate,stat,ssize){
  if(any(stat<0)||any(stat>11)){
    print("Unknown Statistic Requested")
  } else {
    if(nrow(dcsdat)==1){
      print("One value only into timeweightedstats")
      dcsdat <- rbind(dcsdat,dcsdat)
      dcsdat[1,1] <- datestrip(bdate)
      dcsdat[2,1] <- datestrip(edate)
    }

    if(stat==10) {
      if(dcsdat[nrow(dcsdat),1]<datestrip(edate)) {
          dcsdat <- rbind(dcsdat,dcsdat[nrow(dcsdat),])
          dcsdat[nrow(dcsdat),1] <- datestrip(edate)
      }
    }
    tws <- .Call("TimeWeightStats",dcsdat,as.integer(nrow(dcsdat))
        ,
                 as.integer(ncol(dcsdat)), ssize,datestrip(bdate),
                 datestrip(edate), as.integer(stat))
    colnames(tws) <- colnames(dcsdat)[1:2]
    return(tws)

  }
}
```

Listing A.40: `ssldcsutils:getdata` function

```
##' Function to retrieve data from local DCS tables
##'
##' Utility function to retrieve data from local DCS tables.
##' This function is mostly used interactively by
##' supplying the datadesc parameter which is by design human readable.
##'
##'  @param datadesc : Correlates to the \code{DataDescriptor} columname in \code{calctable}
##'  @param bdate : Start date for data download
##'  @param edate : End date for data download
##'  @param calctable : Table in the database that contains the data mapping
##'  @param dbnamedat : dat_ schema in MSPEM structure
##'  @param dbnameapp : app_ schema in MSPEM structure
##'  @param dbtype : Type of database
##'  @param datetime : datetime column name in dat_ schema
##'  @param side : Side of the factory, 0 for data with no side, and NULL if
##'  user wants data from both sides.
##'  @param train : Train to retrieve data for, 0 for data with no train, and NULL if
##'  user wants data from all trains.
##'  @param reactor : Reactor to retrieve data for, 0 for data with no Reactor, and NULL if
##'  user wants data from all Reactors.
##'  @param ssize : Step size for aggregation
##'  @param stat : Statistic for aggregation
##'
##'  @return Dataframe with results.
##'
##'  @export
##'
##


getdata <- function(datadesc,bdate,edate,calctable,dbnamedat,dbnameapp,dbtype,datetime,side=0,train=0,reactor=0,ssize,stat=0) {
    calc <- retrexpcalc(datadesc,calctable,dbnameapp,dbtype,side,train,reactor)
    if(is.null(calc))
        return(NULL)

    tags <- NULL
    for(i in 1:nrow(calc)) {
        tags <- c(tags,parsetags(as.character(calc[i,])))
    }
    tags <- unique(tags)

```

```
42      dbdat <- retrdcsdata(bdate,edate,tags,as.numeric(ssize),as.
            numeric(stat),dbnamedat,dbtype,datetime)
43      colnames(dbdat)[2:ncol(dbdat)] <- paste("C_",gsub("`","",
            colnames(dbdat)[2:ncol(dbdat)]),sep="")
44      dbdat <- as.data.frame(dbdat)
45      res <- dbdat[,1]
46      for(i in 1:nrow(calc)) {
47          res <- cbind(res,with(dbdat,eval(parse(text=calc[i,]))))
48      }
49      colnames(res) <- c(datetime,dataids(datadesc,calctable,
            dbnameapp,dbtype,side,train,reactor))
50      return(as.data.frame(res))
51
52  }
```

Listing A.41: `ssldcsutils:retrexpcalc` function

```
1   ##' Function to retrieve and recursively expand calc given
        datadescriptor
2   ##'
3   ##' Utility function to retrieve and recursively expanded calc
        from \code{calctable}.
4   ##'
5   ##'  @param datadesc : Correlates to the \code{DataDescriptor}
        columnname in \code{calctable}
6   ##'  @param calctable : Table in the database that contains the
        data mapping
7   ##'  @param dbname : app_ schema in MSPEM structure
8   ##'  @param dbtype : Type of database
9   ##'  @param side : Side of the factory, 0 for data with no side,
        and NULL if
10  ##'  user wants data from both sides.
11  ##'  @param train : Train to retrieve data for, 0 for data with no
         train, and NULL if
12  ##'  user wants data from all trains.
13  ##'  @param reactor : Reactor to retrieve data for, 0 for data
        with no Reactor, and NULL if
14  ##'  user wants data from all Reactors.
15  ##'
16  ##'  @return List of expanded calcstrings
17  ##'
18  ##' @export
19
20  retrexpcalc <- function(datadesc,calctable,dbname,dbtype,side=0,
        train=0,reactor=0,calc=NULL) {
21      if(is.null(calc))
22          calc <- retrcalcs(datadesc,calctable,dbname,dbtype,side=
                side,reactor = reactor,train=train)
23
24      if(!is.matrix(calc) & !is.data.frame(calc)) {
25          calc <- matrix(calc,ncol=1)
26      }
27
```

```
    if(nrow(calc)==0)
        return(NULL)
    for(i in 1:nrow(calc)) {
        subclcpos <- getsubcalc(calc[i,])
        while(length(subclcpos)>1){
            subclc <- substr(calc[i,],subclcpos[1],subclcpos[2]+1)
            subclcexp <- unlist(strsplit(gsub("##","",subclc),"#")
                )
        subclcexp <- retrcalcs(subclcexp[1],calctable,dbname,
            dbtype,subclcexp[2],
                                subclcexp[3],subclcexp[4])
            calc[i,] <- gsub(subclc,subclcexp,calc[i,])
            subclcpos <- getsubcalc(calc[i,])
        }
    }
    return(calc)
}
```

Listing A.42: `ssldcsutils:retrcalcs` function

```
##' Function to retrieve calc given datadescriptor
##'
##' Utility function to retrieve calc from \code{calctable} given
    the datadesriptor.
##'
##'  @param datadesc : Correlates to the \code{DataDescriptor}
    columname in \code{calctable}
##'  @param calctable : Table in the database that contains the
    data mapping
##'  @param dbname : app_ schema in MSPEM structure
##'  @param dbtype : Type of database
##'  @param side : Side of the factory, 0 for data with no side,
    and NULL if
##'  user wants data from both sides.
##'  @param train : Train to retrieve data for, 0 for data with no
     train, and NULL if
##'  user wants data from all trains.
##'  @param reactor : Reactor to retrieve data for, 0 for data
    with no Reactor, and NULL if
##'  user wants data from all Reactors.
##'
##'  @return calculation
##'
##' @export
##'

retrcalcs <- function(data,calctable,dbname,dbtype,side=0,train=0,
    reactor=0) {
    ad <- dbconn(dbname,dbtype)
    sqlstr <- paste("SELECT Calculation FROM ",calctable," WHERE
        DataDescriptor = '",datadesc,"'",sep="")
    if(!is.null(side)) {
        sqlstr <- paste(sqlstr," AND Side = '",side,"'",sep="")
```

```
26      }
27      if(!is.null(train)) {
28          sqlstr <- paste(sqlstr," AND Train = '",train,"'",sep="")
29      }
30      if(!is.null(reactor)) {
31          sqlstr <- paste(sqlstr," AND Reactor = '",reactor,"'",sep=
                "")
32      }
33
34      sqlstr <- paste(sqlstr,";",sep="")
35      calc <- sqlQuery(ad,sqlstr)
36      close(ad)
37      return(calc)
38  }
```

Listing A.43: `ssldcsutils:getsubcalc` function

```
1   ##' Function to find sub calc in calc
2   ##'
3   ##' Utility function to find next calculation in the ## format
        inside a current calc. This is used to
4   ##' recursively expand calculation to C_ format.
5   ##'
6   ##' @param calc : Calc string in the ## format
7   ##'
8   ##' @return Next position of calc in the ## format for expansion.
9   ##'
10  ##' @export
11  ##'
12
13  getsubcalc <- function(calc) {
14      gsc <- unlist(gregexpr("##",calc,fixed=TRUE))
15      return(gsc)
16  }
```

Listing A.44: `ssldcsutils:parsetags` function

```
1   ##' Function to retrieve the tags to download from calc in C_
        format.
2   ##'
3   ##' Utility function to extract the tags from a calc string in the
        C_ format.
4   ##'
5   ##' @param calcstring : Calc string in the C_ format
6   ##'
7   ##' @return List of tags to download for calc. Could be empty if
        constant calc string.
8   ##'
9   ##' @export
10
11  parsetags  <- function(calcstring){
12    tags  <- strsplit(calcstring,"\\*|\\+|\\-|\\/|\\(|<|>|\\)|=|\\^"
          )
```

```
  tags <- gsub(" ","",tags[[1]])
        tags <- tags[nzchar(tags)]
  tags <- tags[suppressWarnings(is.na(as.numeric(tags)))]
        tags <- tags[nchar(tags)>1]
  tags  <- unique(tags)
        keep <- rep(TRUE,length(tags))
        for(i in 1:length(tags))
            keep[i] <- (length(grep("_",tags[i])) > 0)
        tags <- tags[keep]
  tags <- substr(tags,3,nchar(tags))
  return(tags)
}
```

Listing A.45: `ssldcsutils:dataids` function

```
##' Utility Function to Return data ids
##'
##' @param datedesc DataDescriptor column of calctable
##' @param calctable cacltable name
##' @param dbnameapp app db name
##' @param dbtype db type
##' @param sep Seperator between different name objects
##'
##' @return vector of DataID's with "#" substituted by \code{sep}
##'
##' @export

dataids <-
function (datadesc,calctable, dbnameapp, dbtype,side=NULL,train=
    NULL,reactor=NULL,sep="._.")
{
    options(stringsAsFactors = FALSE)
    ad <- dbconn(dbnameapp, dbtype)
    sqlstr <- paste("SELECT DataID FROM ", calctable, " WHERE
        DataDescriptor = '",
                    datadesc, "'", sep = "")
    if (!is.null(side)) {
        sqlstr <- paste(sqlstr, " AND Side = '", side, "'", sep =
            "")
    }
    if (!is.null(train)) {
        sqlstr <- paste(sqlstr, " AND Train = '", train, "'",
            sep = "")
    }
    if (!is.null(reactor)) {
        sqlstr <- paste(sqlstr, " AND Reactor = '", reactor,
            "'", sep = "")
    }
    sqlstr <- paste(sqlstr, ";", sep = "")
    tmp <- sqlQuery(ad, sqlstr)
    close(ad)
    dataids <- gsub("#",sep,gsub("##","",tmp$DataID))
    return(dataids)
```

```
}
```

Listing A.46: `ssldcsutils:dataidsfromcalc` function

```
##' Function to change calc in # format to a more dataframe column friendly name
##'
##' @param calc : calc in # format
##' @param sep : seperator to substitute inner #'s with
##'
##' @return calc with outer ##'s remove and inner #'s substituted with \code{sep}
##'
##' @export



dataidsfromcalc <- function(calc,sep = "._.") {
    return(gsub("#", sep, gsub("##", "", calc)))
}
```

Listing A.47: `ssldcsutils:getcacheddata` function

```

##' Function to retrieve and populate local cached tables
##'
##' Utility function to retrieve and populate data from local cached. This function is mostly used interactively by
##' supplying the datadesc parameter which is by design human readable.
##'
##'  @param datadesc : Correlates to the \code{DataDescriptor} columname in \code{calctable}
##'  @param bdate : Start date for data download
##'  @param edate : End date for data download
##'  @param calctable : Table in the database that contains the data mapping
##'  @param dbnamedat : dat_ schema in MSPEM structure
##'  @param dbnameapp : app_ schema in MSPEM structure
##'  @param dbtype : Type of database
##'  @param datetime : datetime column name in dat_ schema
##'  @param side : Side of the factory, 0 for data with no side, and NULL if
##'  user wants data from both sides.
##'  @param train : Train to retrieve data for, 0 for data with no train, and NULL if
##'  user wants data from all trains.
##'  @param reactor : Reactor to retrieve data for, 0 for data with no Reactor, and NULL if
##'  user wants data from all Reactors.
##'  @param ssize : Step size for aggregation
##'  @param stat : Statistic for aggregation
```

```
23 ##'  @param cachpath : Path to the cached data. Currently while
       using \code{SOAR} a directory \code{cachname}
24 ##' will be generated inside cachpath
25 ##'  @param cachname : Name of the cache
26 ##'
27 ##'  @return Dataframe with results.
28 ##'
29 ##'  @export
30 ##'
31 ##
32
33
34 getcacheddata <-  function(datadesc,bdate,edate,calctable,
       dbnamedat,dbnameapp,dbtype,datetime,
35                                side=0,train=0,reactor=0,ssize=15,stat
                                      =0,cachpath="~/rchache",cachname=".R
                                      _Cache") {
36     eval(substitute(Attach(lib.loc=cachpath,lib=cachname),list(
           cachath=cachpath,cachname=cachname)))
37     cname <- .cnameenc(datadesc,ssize,stat)
38     if(exists(cname,envir=as.environment(cachname),inherits=FALSE)
           ) {
39         tmp <- get(cname,envir=as.environment(cachname),
40                                    inherits=FALSE)
41         if(nrow(tmp)>0) {
42         if(max(tmp[,1]) < (datestrip(edate)-ssize*60)) {
43             bdatedat <- dateconvrt(max(tmp[,1]) + ssize*60)
44             edatedat <- edate
45             tmp <- rbind(tmp,getdata(datadesc,bdatedat,edatedat,
                   calctable,dbnamedat,dbnameapp,
46                                dbtype,datetime,NULL,NULL,NULL,ssize,
                                      stat))
47             tmp[order(tmp$DateTime),]
48             eval(substitute({TMP <- tmp; Store(TMP,lib.loc=
                   cachpath,lib=cachname)},
49                                list(TMP=cname,cachpath=cachpath,
                                      cachname=cachname)))
50
51
52         }
53         if(min(tmp[,1]) > (datestrip(bdate))) {
54             bdatedat <- bdate
55             edatedat <- dateconvrt(min(tmp[,1]))
56             tmp <- rbind(getdata(datadesc,bdatedat,edatedat,
                   calctable,dbnamedat,dbnameapp,
57                                dbtype,datetime,NULL,NULL,NULL,ssize,
                                      stat),tmp)
58             tmp[order(tmp$DateTime),]
59             eval(substitute({TMP <- tmp; Store(TMP,lib.loc=
                   cachpath,lib=cachname)},
60                                list(TMP=cname,cachpath=cachpath,
                                      cachname=cachname)))
```

```
61
62          }
63
64          ##Test what happen when assign TMP, and then Store(cname)
65
66          tmp <- as.data.frame(tmp)
67          ##This depends on convention. My convention is label the
                aggregation with start time.
68          ##If that changes then obviously < edate must be <= edate
                and vv.
69          tmp <- tmp[(tmp$DateTime>=datestrip(bdate))&(tmp$DateTime<
                datestrip(edate)),
70                      c(datetime,dataids(datadesc,calctable,dbnameapp
                            ,dbtype,side,train,reactor))]
71      } else {
72          tmp <- getdata(datadesc,bdate,edate,calctable,dbnamedat,
                dbnameapp,
73                                      dbtype,datetime,NULL,NULL,NULL,
                                            ssize,stat)
74
75          eval(substitute({TMP <- tmp; Store(TMP,lib.loc=cachpath,
                lib=cachname)},
76                          list(TMP=cname,cachpath=cachpath,cachname=
                                cachname)))
77          tmp <- tmp[,c(datetime,dataids(datadesc,calctable,
                dbnameapp,dbtype,side,train,reactor))]
78
79      }
80      } else {
81          tmp <- getdata(datadesc,bdate,edate,calctable,dbnamedat,
                dbnameapp,
82                                      dbtype,datetime,NULL,NULL,NULL,
                                            ssize,stat)
83
84          eval(substitute({TMP <- tmp; Store(TMP,lib.loc=cachpath,
                lib=cachname)},
85                          list(TMP=cname,cachpath=cachpath,cachname=
                                cachname)))
86          tmp <- tmp[,c(datetime,dataids(datadesc,calctable,
                dbnameapp,dbtype,side,train,reactor))]
87      }
88      detach(pos=which(search()==cachname))
89      tmp[tmp==-9999] <- NA
90      return(tmp)
91
92  }
```

Listing A.48: `ssldcsutils:cacheutilities` function

```
1
2  ##' @export
3  .cnameenc <- function(datadesc,ssize,stat,sep="._.") {
4      return(paste(datadesc,ssize,stat,sep=sep))
5  }
6
7  ##' @export
8  .cnamedec <- function(cname,sep="._.") {
9      dec <- strsplit(cname,sep,fixed=TRUE)[[1]]
10     dec <- list(DataDescriptor=dec[1],StepSize=as.numeric(dec[2]),
           Statistic=as.numeric(dec[3]))
11     return(dec)
12 }
13
14
15 ##' Utility function to check if obj is a calc
16 ##'
17 ##' @param obj Character name of object
18 ##' @param sep Seperator to search for
19 ##' @export
20 iscalc <- function(obj,sep="._.") {
21     res <- gregexpr(sep,obj,fixed=TRUE)
22     sapply(res,FUN=function(x){all(x>0)})
23 }
24
25 ##' @export
26 .dataidsdec <- function(dataid,sep="._.") {
27     decid <- strsplit(dataid,sep,fixed=TRUE)[[1]]
28     decid <- list(DataDescriptor=decid[1],Side=decid[2],
29                   Train=decid[3],Reactor=decid[4])
30     return(decid)
31 }
```

Listing A.49: `ssldcsutils:cachemaintenance` function

```
##' Function that maintains the cache
##'
##'  @param cachpath : Path to the cached data. Currently while using \code{SOAR} a directory \code{cachname}
##' will be generated inside cachpath
##'  @param cachname : Name of the cache
##'  @param bdate : Start date for data download
##'  @param edate : End date for data download
##'  @param calctable : Table in the database that contains the data mapping
##'  @param dbnamedat : dat_ schema in MSPEM structure
##'  @param dbnameapp : app_ schema in MSPEM structure
##'  @param dbtype : Type of database
##'  @param datetime : datetime column name in dat_ schema
##'
##'
##'  @export

cachemaintenance <- function(cachpath="~/rchache",cachname=".R_Cache",bdate=prevday(edate),
                             edate=roundhour(getcurrtime()),
                                 maxstep=60,minstep=0,calctable,
                             dbnamedat,dbnameapp,dbtype,datetime,
                                 cores=4) {
    eval(substitute(Attach(lib.loc=cachpath,lib=cachname),list(cachath=cachpath,cachname=cachname)))
    cnames = ls(as.environment(cachname))
    if(length(cnames)>0) {
        cnames <- cnames[iscalc(cnames)]


        for(cname in cnames) {
            print(cname)
            dec <- .cnamedec(cname)
            if((dec$StepSize <= maxstep)&(dec$StepSize>=minstep))
                {
                tmp <- get(cname,envir=as.environment(cachname),
                           inherits=FALSE)
                if(nrow(tmp)>0) {
                    rangetmp <- range(tmp$DateTime)


                    if(rangetmp[2]<datestrip(bdate)) {
                        bdate <- dateconvrt(rangetmp[2]+dec$StepSize*60)
                    }

                    if(rangetmp[1] > datestrip(bdate)) {
                        tmp <- NULL
                    } else {
                        tmp <- tmp[tmp$DateTime<datestrip(bdate),]
```

```
                    }
                } else {
                    tmp <- NULL
                }
                tmp <- rbind(tmp, getdata(dec$DataDescriptor, bdate,
                    edate, calctable, dbnamedat, dbnameapp,
                                              dbtype, datetime, NULL, NULL
                                                  , NULL, dec$StepSize, dec
                                                  $Statistic))
                eval(substitute({TMP <- tmp; Store(TMP, lib.loc=
                    cachpath, lib=cachname)},
                                    list(TMP=cname, cachpath=cachpath,
                                        cachname=cachname)))
            }

        }
    }
    detach(pos=which(search()==cachname))
}
```

Listing A.50: `ssldcsutils:parsecalcs` function

```
##' Function to parse calc in the ## format.
##'
##' Utility function to parse calcs in the ## format and return
    the results. This function is used
##' for instance in the dashboard to return the value for the
    dials.
##'
##' @param calc : Calc in the ## format
##'  @param dbnamedat : dat_ schema in MSPEM structure
##'  @param dbnameapp : app_ schema in MSPEM structure
##'  @param dbtype : Type of database
##'  @param datetime : datetime column name in dat_ schema
##'  @param timeinterval : Time in minutes to go back for
    retrieving data.
##'  @param timestep : Stepsize for aggregation
##'  @param stats : Statistic for aggregation
##'  @param returndates : Return the start and end dates with the
    data
##'
##'
##'  @return Either just the numeric results, or with \code{
    returndates} == TRUE,
##'  a list with the results, and the start and end date.
##'
##'  @export
##'
##'

parsecalcs <- function(calc, dbnamedat, dbnameapp, dbtype, datetime,
    calctable, timeinterval=60, timestep=60, stats=0, returndates=FALSE
    ){
```

```
25
26    options(stringsAsFactors=FALSE)
27    calc <- retrexpcalc(Data=NULL,calctable,dbnameapp,dbtype,Side=0,
          Train=0,Reactor=0,calc=calc)
28    tags <- parsetags(as.character(calc))
29    if(length(tags)>0){
30      tags  <- paste("`",tags,"`",sep="")
31      am <- getmaxmax(tags,dbnamedat,dbtype,datetime)
32      am <- getmaxtime(tags[am],dbnamedat,dbtype,datetime)
33
34      sdt <- dateconvrt(datestrip(am)-as.numeric(timeinterval)*60)
35      dbdat <- matrix(0,ceiling(as.numeric(timeinterval)/as.numeric(
            timestep)),length(tags))
36      for(ti in 1:length(tags)){
37        dbdat[,ti] <- retrdcsdata(sdt,am,tags[ti],as.numeric(
              timestep),stats=as.numeric(stats),dbnamedat,dbtype,
              datetime)[,2]
38      }
39
40
41      dbdat[dbdat == -9999] <- NA
42      dbdat[!is.finite(dbdat)] <- NA
43      colnames(dbdat) <- paste("C_",gsub("`","",tags),sep="")
44      dbdat <- as.data.frame(dbdat)
45      res <- with(dbdat,eval(parse(text=calc)))
46    } else {
47      res <- eval(parse(text=calc))
48      am <- sdt <- getcurrtime()
49    }
50    res[!is.finite(res)&!is.character(res)] <- NA
51    if(returndates){
52      return(list(calcval=res,am=am,sdt=sdt))
53    } else {
54      return(res)
55    }
56  }
```

Listing A.51: `ssldcsutils:createdatadescr` function

```
##' Function to retrieve List with unique data descriptors sides reactors and trains
##'
##' Utility function to get a list with all the names for data retrieval
##'
##'  @param calctable : Table in the database that contains the data mapping
##'  @param dbnameapp : app_ schema in MSPEM structure
##'  @param dbtype : Type of database
##'
##' @return List of datadescriptors
##'
##' @export


createdatadescr <- function(calctable,dbnameapp,dbtype) {

    options(stringsAsFactors = FALSE)
    ad <- dbconn(dbnameapp,dbtype)
    sqlstr <- paste("SELECT * FROM `",calctable,"`;",sep="")
    tmp <- sqlQuery(ad,sqlstr)
    close(ad)
    DataDescriptor <- list()

    for(i in unique(tmp$DataDescriptor)) {
        DataDescriptor[[i]] <- i
    }

    return(DataDescriptor)
}
```

Listing A.52: `ssldcsutils:datacrawler` function

```
##' Utility function to test all calcs in calctable
##'
##'  @param dbnamedat : dat_ schema in MSPEM structure
##'  @param dbnameapp : app_ schema in MSPEM structure
##'  @param dbtype : Type of database
##'  @param datetime : datetime column name in dat_ schema
##'  @param timeinterval : Time in minutes to go back for retrieving data.
##'  @param timestep : Stepsize for aggregation
##'  @param stats : Statistic for aggregation
##'
##'  @export

datacrawler <- function(dbnamedat,dbnameapp,dbtype,datetime,calctable,timeinterval=60,timestep=60,stats=0) {
    ad <- dbconn(dbnameapp,dbtype)
```

```
15      calcs <- sqlQuery(ad,paste("SELECT DataID FROM ",calctable,";"
            ,sep=""))
16      close(ad)
17      dc <- list()
18      for(calc in as.vector(unlist(calcs))) {
19          tryCatch(dc[[dataidsfromcalc(calc)]] <- parsecalcs(calc,
                dbnamedat,dbnameapp,dbtype,datetime,calctable,
                timeinterval,timestep,stats),error=function(e) {print(
                calc);print(e)})
20      }
21      return(dc)
22
23  }
```

Listing A.53: `ssldcsutils:tagexist` function

```
1   ##' Utility Function to check if tags in database
2   ##'
3   ##' Small wrapper function to check if new tags already exists in
        database
4   ##'
5   ##' @param taglist New tags that needs to be checked
6   ##' @param tagtable Table in the database that contains the
        existing tag list
7   ##' @param tagtablecol Column in tagtable that contains the list
        of tagnames
8   ##' @param dbname Database with tags
9   ##' @param dbtype Type of database
10  ##'
11  ##' @return A vector of boolean values of the same length as \code
        {taglist}.
12  ##'
13  ##' @export
14
15  tagexist <- function(taglist,tagtable,tagtablecol,dbname,dbtype) {
16      ad <- dbconn(dbname,dbtype)
17      sqlstr <- paste("SELECT `",tagtablecol,"` FROM `",tagtable,"`;
            ",sep="")
18      exsttags <- sqlQuery(ad,sqlstr)
19      close(ad)
20      tagexist <- as.vector(unlist(taglist)) %in% as.vector(unlist(
            exsttags))
21      names(tagexist) <- as.vector(unlist(taglist))
22      return(tagexist)
23  }
```

A.5 Excel Interface Functions (sslxlutils)

Listing A.54: sslxlutils::trawlxls4data function

```
##' Function to grab Excel data form spreadsheets on servers and dump into
##' fake tags for further use
##'
##' @param xlstrawl Table with input for trawler
##' @param appname Name of R application for error logger
##' @param dbnameapp Database for app
##' @param dbnamedat Database for dat
##' @param dbtype Type of database
##' @param datetime datetime column
##' @param logtable Name of table in dbnameapp to log errors to.
##' @param javapars Parameters to send to \code{options(java.parameters)}
##' @param numhistdays Number of days to go back every time to make sure data
##' is current
##'
##' @export


trawlxls4data <- function(xlstrawl,appname,dbnameapp,dbnamedat,dbtype,datetime,logtable,
                          numhistdays=2) {

    ##Get unique file ids to ensure each xls file is opened only once
    xlsworkbookids <- unique(xlstrawl$FileIdentifier)

    for(wkb in xlsworkbookids) {

        filenameraw <- xlstrawl[xlstrawl$FileIdentifier==wkb,"FileName"][1]
        if(xlstrawl[xlstrawl$FileIdentifier==wkb,"RawFileName"][1]==1) {
            ##Get max time. Assumption is dat al die data uit een excel sheet ten minste selfde dag sal wees.
            md <- getmaxtime(xlstrawl[xlstrawl$FileIdentifier==wkb,"Tag"][1],dbnamedat,dbtype,datetime)
            fileday <- datestrip(roundday(md))-numhistdays*24*3600
            endday <- datestrip(roundday(getcurrtime()))-
                                    xlstrawl[xlstrawl$
                                        FileIdentifier==wkb,"
                                        NumDaysOffset"][1]*24*3600
            xlsbookdat <- xlstrawl[xlstrawl$FileIdentifier==wkb,]

            while(fileday <= endday) {
                print(dateconvrt(fileday))
```

```
37                  filename <-  dateconvrt(fileday,format=filenameraw
                        )
38                  if(file.exists(filename)) {
39                      wb <- loadWorkbook(filename)
40                      xlstrawlworker(wb,xlsbookdat,fileday,dbnamedat
                            ,dbtype,datetime)
41                      rm(wb)
42                  } else {
43                      err <- paste(filename,"does not exist.")
44                      print(err)
45                      errlog(dbnameapp,dbtype,logtable,appname,"
                            trawlxls4data",err)
46
47                  }
48
49                  fileday <- datestrip(nextday(dateconvrt(fileday)))
50              }
51
52          } else {
53              if(file.exists(filenameraw)) {
54                  wb <- loadWorkbook(filenameraw)
55                  xlsbookdat <- xlstrawl[xlstrawl$FileIdentifier==
                        wkb,]
56                  xlstrawlworker(wb,xlsbookdat,datestrip(roundday(
                        getcurrtime())),dbnamedat,dbtype,datetime)
57
58                  rm(wb)
59              } else {
60                  err <- paste(filenameraw,"does not exist.")
61                  print(err)
62                  errlog(dbnameapp,dbtype,logtable,appname,"
                        trawlxls4data",err)
63              }
64
65          }
66
67
68
69
70
71      }
72
73  }
```

Listing A.55: `sslxlutils::xlstrawlworker` function

```
1
2  ##' Helper function to get data form open XLConnect workbook
3  ##'
4  ##' @param wb XLConnect connector
5  ##' @param xlstrawldat Table with trawler data for specific
       workbook
6  ##' @param curday Current day for date calcs
7  ##' @param dbnamedat Database for dat
8  ##' @param dbtype Type of database
9  ##' @param datetime datetime column
10 ##'
11 ##' @export
12 ##'
13
14 xlstrawlworker <- function(wb, xlstrawldat, curday, dbnamedat, dbtype,
       datetime) {
15
16     for(i in 1:nrow(xlstrawldat)) {
17         ##Data should be relatively uncomplicated. Why read the
               sheet if you do not
18         ##want the data?
19         dat <- read.xlsx(wb, sheet=xlstrawldat$SheetName[i],
20                          rows=c(xlstrawldat$StartRowData[i]:
                                 xlstrawldat$EndRowData[i]),
21                          cols=c(xlstrawldat$StartColData[i]:
                                 xlstrawldat$EndColData[i]),
22                          skipEmptyRows = FALSE, colNames = FALSE)
23
24         if(xlstrawldat$DateType[i]=="CURRENT") {
25             date <- dateconvrt(curday, format="%d-%b-%y")
26         } else {
27             if(xlstrawldat$DateType[i]=="READ") {
28                 date <- read.xlsx(wb, sheet=xlstrawldat$SheetName[i
                       ],
29                                   rows=c(xlstrawldat$StartRowDate[
                                       i]:xlstrawldat$EndRowDate[i])
                                       ,
30                                   cols=c(xlstrawldat$StartColDate[
                                       i]:xlstrawldat$EndColDate[i])
                                       ,
31                                   skipEmptyRows = FALSE, colNames=
                                       FALSE)
32
33             } else {
34                 ##This is unlikely to happen
35                 if(xlstrawldat$DateType[i]=="FIXED") {
36                     date <- xlstrawldat$DateValue[i]
37                 }
38
39             }
40
```

```
41          }
42
43          if(xlstrawldat$TimeType[i]=="CURRENT") {
44              time <- dateconvrt(curday,format="%d-%b-%y")
45          } else {
46              if(xlstrawldat$TimeType[i]=="READ") {
47                  time <- read.xlsx(wb,sheet=xlstrawldat$SheetName[i
                        ],
48                                    rows=c(xlstrawldat$StartRowTime[
                                        i]:xlstrawldat$EndRowDate[i])
                                        ,
49                                    cols=c(xlstrawldat$StartColTime[
                                        i]:xlstrawldat$EndColTime[i])
                                        ,
50                                    skipEmptyRows = FALSE,colNames=
                                        FALSE)
51
52              } else {
53                  ##This is unlikely to happen
54                  if(xlstrawldat$TimeType[i]=="FIXED") {
55                      time <- xlstrawldat$TimeValue[i]
56                  } else {
57                      time <- NULL
58                  }
59
60              }
61
62          }
63          if(length(dat)>0) {
64              tagvals <- createtagdatafromxls(date,time,dat,status
                    =0,xlstrawldat$DateFormat[i],
65                                              xlstrawldat$TimeFormat
                                                    [i])
66              tagvals <- na.omit(tagvals,drop=FALSE)
67              if(nrow(tagvals)>0) {
68                  exptodb(tagvals,xlstrawldat$Tag[i],dbnamedat,
                        dbtype,TRUE)
69              }
70          }
71      }
72  }
```

Listing A.56: `sslxlutils::createtagdatafromxls` function

```
##' Function to create Tag data from xls data
##'
##' Utility function to encapsulate the creation of a dim n x 3 table with
##' colnames = c("DateTime","Value","Status")
##'
##' @param date Date for data as character values, possible including time
##' values if \code{is.null(time)}
##' @param time Time for data, if \code{NULL} then included in datetime
##' @param dat Vector of data values
##' @param status Vector of status values. If \code{length(status)==1} will be
##' expanded to \code{status <- rep(status,length(dat))}
##' @param dateformat Format of the date values
##' @param timeformat Format of the time values
##'
##' @return dim n x 3 dataframe with \code{colnames = c("DateTime","Value","Status")}
##'
##' @export
##'

createtagdatafromxls <- function(date,time=NULL,dat,status=0,dateformat="%d-%b-%y",
                                 timeformat="%H:%M") {
    ##Current assumption is that length(dat) is law. date, time and status will be
    ##expanded/truncated to be ==length(dat).
    if(is.data.frame(dat)) {
        dat <- c(unlist(dat))
    }
    if(is.data.frame(date)) {
        date <- c(unlist(date))
    }
    if(is.data.frame(time)) {
        time <- c(unlist(time))
    }
    assert_that(length(dat)>=1)
    ##For date/time only contraction make sense, else duplicate values will be
    ##introduced


    if(is.null(time)) {

        assert_that(length(dat) <= length(date))
        date <- date[1:length(dat)]
        if(is.numeric(date)) {
           date <- datestrip(exceldateconvrt(date))
```

```
44          } else {
45                  date <- datestrip(date,format=dateformat)
46            }
47      } else {
48          ##Do not make sense to have more dates than times...
49          assert_that(length(date) <= length(time))
50          if(length(date)<length(time)) {
51              ##Can currently only handle length(date)==1
52              assert_that(length(date)==1)
53              date <- rep(date,length(time))
54          }
55
56          date <- paste(date,time,sep=" - ")
57          dateformat <- paste(dateformat,timeformat,sep=" - ")
58          date <- datestrip(date,format=dateformat)
59      }
60
61      if(length(status)>1) {
62          ##status must be at least as long as dat, else need to
                make assumptions
63          ##about status
64          assert_that(length(status)>=length(dat))
65          status <- status[1:length(dat)]
66      } else {
67          assert_that(length(status)==1)
68          status <- rep(status,length(dat))
69      }
70
71      assert_that(are_equal(length(dat),length(date)))
72      assert_that(are_equal(length(dat),length(status)))
73
74      tagdat <- cbind(date,dat,status)
75      colnames(tagdat) <- c("DateTime","Value","Status")
76      rownames(tagdat) <- 1:nrow(tagdat)
77      return(as.data.frame(tagdat))
78  }
```

Appendix B

Statistical Software

B.1 Code Listings for the `mltv` package

B.1.1 PCA and CVA Algebra

Listing B.1: `mltv::mltvcva` function

```
##' CVA Function
##'
##' @param X  Data Matrix
##' @param g  Vector of groups of length nrow(x)
##' @param evec  Eigenvectors to use for scaffolding.
##' @param weightedCVA  Weights to use for CVA, eihter "weighted","unweightedI" or "unweightedCent"
##'
##' @return Object of class c(mltv_cva,mltv) containing the following output:
##' \itemize{
##' \item \code{data} - Original input data (\code{X}).
##' \item \code{evec} - Eigenvectors to include.
##' \item \code{colmeans} - The mean value for each column of \code{X}.
##' \item \code{colsd} - The standard deviation for each column of \code{X}.
##' \item \code{groups} - The groups assigned to each row of \code{X}.
##' \item \code{axes} - Variable axes to include in biplot.
##' \item \code{axnames} - Names of variable axes.
##' \item \code{vraxes} - Loadings to use for axes marker projections.
##' \item \code{vrpoints} - Loadings to use for new sample projections.
##' \item \code{projmean} - Projected mean values.
##' \item \code{projpoints} - Projected values for samples (rows of \code{X}).
##' \item \code{quality} - CVA quality in the original variables.
```

```
22 ##' \item \code{classification} - Classification prediction
       accuracy.
23 ##' \item \code{transformmat} - Transformation matrix for original
       \code{X} values
24 ##' to the reduced canonical space.
25 ##' \item \code{transformmatinv} - Transformation matrix form
       reduced canonical space
26 ##' to original space.
27 ##' \item \code{weights} - CVA weights for groups.
28 ##' \item \code{scaled} - Always \code{FALSE} but included for
       axis calculations.
29 ##' \item \code{datcent} - The centered \code{X} data.
30 ##' }
31 ##'
32 ##' @export
33 ##'
34 ##'
35 ##'
36
37
38
39 mltvcva <- function(X,g,weightedCVA = c("weighted"),evec = c(1,2))
       {
40     cvabipl <- list()
41     ##Setup some initial variables
42     X <- as.matrix(X)
43     n <- nrow(X)
44     p <- ncol(X)
45
46     cvabipl[["data"]] <- X
47     cvabipl[["groups"]] <- g
48     cvabipl[["evec"]] <- evec
49     ## G is a j x n indicator matrix with j = number of groups
           with
50     ## 1's in the j_i th columns of G for each n_i in group j_i
           and zeros otherwise
51     G <- gmembermat(g)
52     j <- ncol(G)
53     ##N is a  pxp diagonal matrix with the size of the groups on
           each diagonal
54     N <- t(G)%*%G
55     Xbarall <- colMeans(X)
56     Xsdall <- apply(X,2,sd)
57     cvabipl[["colmeans"]] <- Xbarall
58     cvabipl[["colsd"]] <- Xsdall
59     Xcent <- scale(X,center=T,scale=F)
60     Xbargrp <- solve(N)%*%t(G)%*%X
61     Xsdgrp <- summaryBy(.~g,data=cbind(as.data.frame(X),g),keep.
           names = T,FUN=sd)
62     Xsdgrp <- Xsdgrp[,-1]
63   ##Get scaled group means
64     Xcentbargrp <- solve(N)%*%t(G)%*%Xcent
```

```
65      Xcentsdgrp <- summaryBy(.~g,data=cbind(as.data.frame(Xcent),g)
            ,keep.names = T,FUN=sd)
66      Xcentvargrp <- summaryBy(.~g,data=cbind(as.data.frame(Xcent),g
            ),keep.names = T,FUN=var)
67      Xcentsdgrp <- Xcentsdgrp[,-1]
68      Xcentvargrp <- Xcentvargrp[,-1]
69    ##Calculates the SSP matrices
70      SSP.T <- t(Xcent)%*%Xcent
71      SSP.B <- t(Xcentbargrp)%*%N%*%Xcentbargrp
72      SSP.W <- SSP.T - SSP.B
73
74      ##Calculates the transformation matrix L that transforms the
            original variables to the
75      ##canonical variables in the canonical space (4.4) page 152 in
             [Gower 2011]
76
77      W <- SSP.W
78      W.svd <- svd(W)
79      W.lambda <- diag(W.svd$d)
80
81      L <- W.svd$u %*% solve(sqrt(W.lambda))
82
83      ## Second phase - dimension reduction
84      ## Consider three choices for C: 1. Weighted: N; 2. Unweighted
            : I-(11')/J and 3. I
85      if(weightedCVA == "weighted")
86          {
87              C <- N
88              Csqrt <- sqrt(N)
89          }
90      if(weightedCVA == "unweightedI")
91          {
92              C <- diag(j)
93              Csqrt <- C
94          }
95      if(weightedCVA == "unweightedCent")
96          {
97              C <- diag(j) - matrix(1/j, nrow=j,ncol=j)
98              Csqrt <- C
99          }
100     svdLxcentbargrp <- svd(t(L) %*% t(Xcentbargrp) %*% C %*%
            Xcentbargrp %*% L)
101
102     V <- svdLxcentbargrp$v
103     Lambda <- diag(svdLxcentbargrp$d)
104
105     M <- L%*%V
106
107     Minv <- solve(M)
108
109   ##Create J matrix to facilitate ease of calculation
110     J <- rep(0,p)
```

```
111     J[evec] <- 1
112     J <- diag(J)
113
114   ## Do the conversion to canonical and dimension reduction in one
          step
115     Xcentbarcanred  <- Xcentbargrp %*% M %*% J
116     Xcentcanred     <- Xcent %*% M %*% J
117
118   ## Calculates X^hat (X^bar)^hat
119     Xcentbarhat <-  Xcentbarcanred %*% Minv
120     Xcenthat <- Xcentcanred %*% Minv
121
122     Xcent.ssq <- sum((Xcent - Xcenthat)^2)
123     Xcentbar.ssq <- sum((Xcentbargrp - Xcentbarhat)^2)
124     MJMI <-  M %*% J %*% Minv
125     I <- diag(rep(1,p))
126     Sigma <- cov(Xbargrp)
127     Vexp <- t(I - MJMI) %*% Sigma %*% (I - MJMI)
128     grpweights <- colSums(G)/sum(colSums(G))
129
130
131     ##Quality in both canonical and original variables
132     lambdavec <- zapsmall(diag(Lambda))
133     CVAqualitycanvar <- sum(lambdavec[evec])/sum(lambdavec)
134     CVAqualityorigvar <- sum(diag(t(Xcentbarhat) %*% C %*%
            Xcentbarhat)) / sum(diag(t(Xcentbargrp) %*% C %*%
            Xcentbargrp))
135
136   ###Adequacy
137
138     adequacy <- diag(M[,evec] %*% t(M[,evec]))/diag(M %*% t(M))
139
140     names(adequacy) <- colnames(X)
141
142
143   Xcentbarpoints <- Xcentbargrp%*%M[,evec]
144   Xcentpoints <- Xcent%*%M[,evec]
145
146   eucldist <- function(xcp,xcbp) {
147     eucl <- function(xcpi,xcp,xcbp) {
148       xcpimat <- matrix(rep(xcp[xcpi,],nrow(xcbp)),ncol=2,byrow=T)
149       eucl <- sqrt(rowSums((xcbp-xcpimat)^2))
150       return(eucl)
151     }
152     eucldist <- sapply(1:nrow(xcp),FUN=eucl,xcp=xcp,xcbp=xcbp)
153     return(t(eucldist))
154
155   }
156   cvaclasspred <- function(Xcentpoint,Xcentbarpoints) {
157     Xcentpointsclass <- eucldist(Xcentpoints,Xcentbarpoints)
158     predclass <- apply(Xcentpointsclass,FUN=which.min,MARGIN=1)
159     return(predclass)
```

```
160   }
161
162   Xcentpointspredclassnr <- cvaclasspred(Xcentpoint,Xcentbarpoints
          )
163   Xcentpointspredclass <- rownames(Xcentbarpoints)[
          Xcentpointspredclassnr]
164   cvaclasspredperc <- NULL
165   for(i in unique(g)) {
166     cvaclasspredperc[[i]] <- sum(g[(g==i)]==Xcentpointspredclass[g
          ==i])/sum(g==i)
167   }
168
169     axnames <- colnames(X)
170
171     axes <- 1:p
172
173     cvabipl[["axes"]] <- axes
174     cvabipl[["axnames"]] <- axnames
175     cvabipl[["vraxes"]] <- t(Minv[evec,])
176     cvabipl[["vrpoints"]] <- M[,evec]
177     cvabipl[["projmean"]] <- Xcentbarpoints
178     cvabipl[["projpoints"]] <- Xcentpoints
179     cvabipl[["quality"]] <- CVAqualityorigvar * 100
180     cvabipl[["classification"]] <- cvaclasspredperc
181     cvabipl[["transformmat"]] <- M
182     cvabipl[["transformmatinv"]] <- Minv
183     cvabipl[["scaled"]] <- FALSE
184     cvabipl[["weights"]] <- C
185     cvabipl[["datcent"]] <- Xcent
186     class(cvabipl) <- c("mltv_cva","mltv")
187     return(cvabipl)
188
189 }
```

Listing B.2: `mltv::mltvpca` function

```
##' PCA Function
##'
##' @param X Data Matrix
##' @param g Optional grouping variable
##' @param evec Eigenvectors to include.
##'
##' @return Object of class c(mltv_pca,mltv) containing the
    following output:
##' \itemize{
##' \item \code{data} - Original input data (\code{X}).
##' \item \code{evec} - Eigenvectors to include.
##' \item \code{colmeans} - The mean value for each column of \
    code{X}.
##' \item \code{colsd} - The standard deviation for each column of
     \code{X}.
##' \item \code{groups} - The groups assigned to each row of \code
    {X}.
##' \item \code{axes} - Variable axes to include in biplot.
##' \item \code{axnames} - Names of variable axes.
##' \item \code{vraxes} - Loadings to use for axes marker
    projections.
##' \item \code{vrpoints} - Loadings to use for new sample
    projections.
##' \item \code{projmean} - Projected mean values.
##' \item \code{projpoints} - Projected values for samples (rows
    of \code{X}).
##' \item \code{quality} - PCA quality.
##' \item \code{scaled} - Corresponds to the \code{scalemat}
    parameter.
##' \item \code{eigen} - Eigenvectors corresponding to the \code{
    evec} parameter.
##' \item \code{sigma} - All the calculated eigenvalues.
##' \item \code{loadings} - All the calculated loadings.
##' \item \code{datcent} - The centered \code{X} data.
##' }
##'
##'
##' @export
##'


mltvpca <- function(X,g=NULL,evec=c(1,2),scalemat=TRUE) {

  pcabipl <- list()
  #obtain biplot scaffolding
  n <- nrow(X)
  p <- ncol(X)

  pcabipl[["data"]] <- X
  pcabipl[["evec"]] <- evec

```

```
43  Xbarall <- apply(X,2,mean)
44  Xsdall <- apply(X,2,sd)
45  pcabipl[["colmeans"]] <- Xbarall
46  pcabipl[["colsd"]] <- Xsdall
47
48  #Center and scale X
49
50  Xcent <- scale(X, center=TRUE, scale=scalemat)
51
52  if(is.null(g)) {
53      g <- rep("mean",nrow(X))
54  }
55  pcabipl[["groups"]] <- g
56
57  ## G is a j x n indicator matrix with j = number of groups with
58  ## 1's in the j_i th columns of G for each n_i in group j_i and
        zeros otherwise
59  G <- gmembermat(g)
60  j <- ncol(G)
61  ##N is a  pxp diagonal matrix with the size of the groups on
        each diagonal
62  N <- t(G)%*%G
63
64  Xbargrp <- solve(N)%*%t(G)%*%X
65  Xsdgrp <- summaryBy(.~g,data=cbind(as.data.frame(X),g),keep.
        names = T,FUN=sd)
66  Xsdgrp <- Xsdgrp[,-1]
67  ##Get scaled group means
68  Xcentbargrp <- solve(N)%*%t(G)%*%Xcent
69  Xcentsdgrp <- summaryBy(.~g,data=cbind(as.data.frame(Xcent),g),
        keep.names = T,FUN=sd)
70  Xcentvargrp <- summaryBy(.~g,data=cbind(as.data.frame(Xcent),g),
        keep.names = T,FUN=var)
71  Xcentsdgrp <- Xcentsdgrp[,-1]
72  Xcentvargrp <- Xcentvargrp[,-1]
73
74  Xcentsvd <- svd(Xcent)
75  Vr <- Xcentsvd$v[,evec]
76  Xcentpoints <- Xcent %*% Vr
77  Xcentbarpoints <- Xcentbargrp%*%Vr
78  Xhat <- X%*%Vr%*%t(Vr)
79  axnames <- colnames(X)
80
81  Xcenthat <- Xcent%*%Vr%*%t(Vr)
82
83  ##Calculates quality metric
84
85  quality <- (sum(Xcentsvd$d[evec]^2)/sum(Xcentsvd$d^2))*100
86
87  axes <- 1:p
88
89  pcabipl[["axes"]] <- axes
```

```
90    pcabipl[["axnames"]] <- axnames
91    pcabipl[["vraxes"]] <- Vr
92    pcabipl[["vrpoints"]] <- Vr
93    pcabipl[["projmean"]] <- Xcentbarpoints
94    pcabipl[["projpoints"]] <- Xcentpoints
95    pcabipl[["quality"]] <- quality
96    pcabipl[["scaled"]] <- scalemat
97    pcabipl[["eigen"]] <- Xcentsvd$d[evec]
98    pcabipl[["sigma"]] <-  Xcentsvd$d
99    pcabipl[["loadings"]] <- Xcentsvd$v
100   pcabipl[["datcent"]] <- Xcent
101   class(pcabipl) <- c("mltv_pca","mltv")
102   return(pcabipl)
103
104 }
```

B.1.2 Data projection functions

Listing B.3: `mltv::project` function

```
##' Generic method to project new data into r_* dimensions
##'
##' @param mltv Object of one of the \code{mltv} classes.
##' @param xnew New data to project.
##' @param g Optional grouping variable.

##' @return xnew projected to r_* dimensions.
##'
##' @export
##'

project <- function(mltv,xnew,g) {
    UseMethod("project")
}
```

Listing B.4: `mltv::project.mltv-pca` function

```
##' Method that returns projected values for PCA
##'
##' @param mltv_pca Output from \code{mltvpca} function.
##' @param xnew Data matrix to project (unscaled).
##' @param g Optional grouping variable.
##'
##' @return \code{mltv_pca} object with added xnew slot containing
    :
##' \itemize{
##' \item \code{data} - Original input data (\code{xnew}).
##' \item \code{groups} - The groups assigned to each row of \code
    {xnew}.
##' \item \code{colmeans} - The mean value for each column of \
    code{xnew}.
##' \item \code{colsd} - The standard deviation for each column of
     \code{xnew}.
##' \item \code{projpoints} - Projected values for samples (rows
    of \code{xnew}).
##' \item \code{datcent} - The centered \code{xnew} data.
##' }
##'
##' @export
##'

project.mltv_pca <- function(mltv_pca,xnew,g=NULL) {

    if(mltv_pca$scaled) {
        xnewcent <- scale(xnew,center=mltv_pca$colmeans,scale=mltv
            _pca$colsd)
    } else {
        xnewcent <- scale(xnew,center=mltv_pca$colmeans,scale=
            FALSE)
```

```
26      }
27      xnewcentproj <- xnewcent%*%mltv_pca$vrpoints
28      xnewmeanproj <- colMeans(xnewcent)%*%mltv_pca$vrpoints
29
30      xnewproj <- list()
31      xnewproj[["data"]] <- xnew
32      xnewproj[["groups"]] <- ifelse(is.null(g),rep("xnew",nrow(xnew
            )),g)
33      xnewproj[["colmeans"]] <- colMeans(xnew)
34      xnewproj[["colsd"]] <- apply(xnew,2,sd)
35      xnewproj[["projpoints"]] <- xnewcentproj
36      xnewproj[["datcent"]] <- xnewcent
37
38      mltv_pca[["xnew"]] <- xnewproj
39
40      return(mltv_pca)
41
42  }
```

B.1.3 Axis measures of fit

Listing B.5: `mltv::axispredictivity` function

```
##' Generic function for axis predicitivity methods
##'
##' @param mltv Output form one of the mltv functions.
##' @param axlim Cutoff value for the inclusions of axis in the
    biplot.
##'
##' @export
##'

axispredictivity <- function(mltv,axlim) {
    UseMethod("axispredictivity")
}
```

Listing B.6: `mltv::axispredictivity.mltv-pca` function

```
##' Method to calculate Axis predictivity for the mltv_pca class
##'
##' @param mltv_pca Output from the mltvpca Function
##' @param axlim Cutoff point for axis. \code{NULL} indicates no
    cutoff.
##'
##' @return Object of class \code{axispred_predictivity} containg:
##' \itemize{
##' \item \code{axispred} - Calculated axis predictivity values.
##' \item \code{axispredlimit} - Cutoff point for axis inclusion.
##' }
##' The axispred object is inserted in a slot named "axispred" in
    the \code{mltv_pca} object,
##' and the \code{mltv_pca} object is returned.
##' @export
##'
axispredictivity.mltv_pca <- function(mltv_pca,axlim=NULL) {

    Xcent <- mltv_pca$datcent
    Vr <- mltv_pca$vrpoints
    Xcenthat <- Xcent%*%Vr%*%t(Vr)

    axispredictivity <- diag(t(Xcenthat)%*%Xcenthat)/diag(t(Xcent)
        %*%Xcent)

    axispred <- list()
    axispred[["axispred"]] <- axispredictivity
    axispred[["axispredlimit"]] <- axlim
    class(axispred) <- c("axispred_predictivity","axispred")
    mltv_pca[["axispred"]] <- axispred

    return(mltv_pca)
}
```

Listing B.7: `mltv::axispredictivity.mltv-cva` function

```
##' Method to calculate Axis predictivity for the mltv_cva class
##'
##' @param mltv_cva Output from the mltvcva Function
##' @param axlim Cutoff point for axis. \code{NULL} indicates no cutoff.
##'
##'
##' @return Object of class \code{axispred_predictivity} containg:
##' \itemize{
##' \item \code{axispred} - Calculated axis predictivity values.
##' \item \code{axispredlimit} - Cutoff point for axis inclusion.
##' }
##' The axispred object is inserted in a slot named "axispred" in the \code{mltv_cva} object,
##' and the \code{mltv_cva} object is returned.

##' @export
##'


axispredictivity.mltv_cva <- function(mltv_cva,axlim=NULL) {

    J <- rep(0,ncol(mltv_cva$data))
    J[mltv_cva$evec] <- 1
    J <- diag(J)
    G <- gmembermat(mltv_cva$groups)
    N <- t(G)%*%G
    Xcentbargrp <- solve(N)%*%t(G)%*%mltv_cva$datcent
    Xcentbarhat  <- Xcentbargrp %*% mltv_cva$transformmat %*% J %*% mltv_cva$transformmatinv


    axispredictivity <- diag(t(Xcentbarhat) %*% mltv_cva$weights %*% Xcentbarhat) /
    diag(t(Xcentbargrp) %*% mltv_cva$weights %*% Xcentbargrp)


    axispred <- list()
    axispred[["axispred"]] <- axispredictivity
    axispred[["axispredlimit"]] <- axlim
    class(axispred) <- c("axispred_predictivity","axispred")
    mltv_cva[["axispred"]] <- axispred

    return(mltv_cva)

}
```

Listing B.8: `mltv::axismspe` function

```
##' Generic function for Axis mspe values
##'
##' @param mltv Output form one of the mltv functions.
##' @param axlim Cutoff value for the inclusions of axis in the
    biplot.
##'
##' @export
##'

axismspe <- function(mltv,axlim) {
    UseMethod("axismspe")
}
```

Listing B.9: `mltv::axismspe.mltv-pca` function

```
##' Method to calculate Axis predictivity for the mltv_pca class
##'
##' @param mltv_pca Output from the mltvpca Function
##' @param axlim Cutoff point for axis. \code{NULL} indicates no
    cutoff.
##'
##' @return Object of class \code{axispred_mspe} containg:
##' \itemize{
##' \item \code{axispred} - Calculated axis mspe values.
##' \item \code{axispredlimit} - Cutoff point for axis inclusion.
##' }
##' The axispred object is inserted in a slot named "axispred" in
    the \code{mltv_pca} object,
##' and the \code{mltv_pca} object is returned.
##'
##' @export
##'


axismspe.mltv_pca <- function(mltv_pca,axlim=NULL) {

    Xcent <- mltv_pca$datcent
    Vr <- mltv_pca$vrpoints
    Xcenthat <- Xcent%*%Vr%*%t(Vr)

    axismspe <- colSums(abs(Xcent - Xcenthat))/N
    axispred <- list()
    axispred[["axispred"]] <- axismspe
    axispred[["axispredlimit"]] <- axlim
    class(axispred) <- c("axispred_mspe","axispred")

    mltv_pca[["axispred"]] <- axispred

    return(mltv_pca)

}
```

Listing B.10: `mltv::axismspe.mltv-cva` function

```
##' Method to calculate Axis mspe for the mltv_cva class
##'
##' @param mltv_cva Output from the mltvcva Function
##' @param axlim Cutoff point for axis. \code{NULL} indicates no cutoff.
##'
##'
##' @return Object of class \code{axispred_mspe} containg:
##' \itemize{
##' \item \code{axispred} - Calculated axis mspe values.
##' \item \code{axispredlimit} - Cutoff point for axis inclusion.
##' }
##' The axispred object is inserted in a slot named "axispred" in the \code{mltv_cva} object,
##' and the \code{mltv_cva} object is returned.

##' @export
##'


axismspe.mltv_cva <- function(mltv_cva,axlim=NULL) {

    J <- rep(0,ncol(mltv_cva$data))
    J[mltv_cva$evec] <- 1
    J <- diag(J)
    G <- gmembermat(mltv_cva$groups)
    N <- t(G)%*%G
    Xcentbargrp <- solve(N)%*%t(G)%*%mltv_cva$datcent
    Xcentbarhat  <- Xcentbargrp %*% mltv_cva$transformmat %*% J %*% mltv_cva$transformmatinv


    sdgrp <- apply(Xcentbargrp,2,sd)
    axismspe <- (rowMeans((abs(Xcentbargrp-Xcentbarhat))))/sdgrp)

    axispred <- list()
    axispred[["axispred"]] <- axismspe
    axispred[["axispredlimit"]] <- axlim
    class(axispred) <- c("axispred_mspe","axispred")
    mltv_cva[["axispred"]] <- axispred

    return(mltv_cva)

}
```

Listing B.11: `mltv::axisinclude` function

```
##' Generic function to calculate axis inclusion
##'
##' @param axispred Output form one of the axispred functions.
##'
##' @export
##'

axisinclude <- function(axispred) {
    UseMethod("axisinclude")
}
```

Listing B.12: `mltv::axisinclude.axispred-predictivity` function

```
##' Method to calculate axis inclusion for axis predictivity.
##'
##' @param axispred Output form one of the axispred functions.
##'
##' @return Numeric vector of axis to include in biplot.
##'
##' @export
##'

axisinclude.axispred_predictivity <- function(axispred) {
    return((1:length(axispred$axispred))[axispred$axispred >=
        axispred$axispredlimit])
}
```

Listing B.13: `mltv::axisinclude.axispred-mspe` function

```
##' Method to calculate axis inclusion for mspe.
##'
##' @param axispred Output form one of the axispred functions.
##'
##' @return Numeric vector of axis to include in biplot.
##'
##' @export
##'

axisinclude.axispred_mspe <- function(axispred) {
    return((1:length(axispred$axispred))[axispred$axispred <=
        axispred$axispredlimit])
}
```

B.1.4 Data enclosure functions

Listing B.14: `mltv::createbags` function

```
##' Function to create list of input for alphabags.
##'
##' @param mltv : Object of class \code{mltv}.
##' @param alpha : alpha value for \code{alphabag} function.
##'
##' @return list with groupnames in \code{mltv$groups} containing
    coordinates for alphabags
##' for the groups.  The list is added to the "container" slot in
    the \code{mltv} object.
##'
##' @export

createbags <- function(mltv,alpha=0.90) {
    assert_that(inherits(mltv,"mltv"))
    bags <- list()
    for(grp in unique(mltv$groups)) {
        bags[[grp]] <- alphabag(mltv$projpoints[mltv$groups==grp
            ,1],
                                mltv$projpoints[mltv$groups==grp,2])
    }
    mltv[["container"]] <- bags
    return(mltv)
}
```

Listing B.15: `mltv::alphabag` function

```
##' Function to return alphabag points for plotting
##' It does not plot the alphabag.
##'
##' @param x : x values
##' @param y : y values
##' @param alpha : alpha value for the bag
##'
##' @export

alphabag <- function (x,y, alpha = 0.9)
{
    n <- length(x)
    storage.mode(x) <- "double"
    storage.mode(y) <- "double"
    interpx <- rep(0, 2 * n)
    storage.mode(interpx) <- "double"
    interpy <- rep(0, 2 * n)
    storage.mode(interpy) <- "double"
    datatyp <- matrix(0, n, 3)
    storage.mode(datatyp) <- "double"
    datatyp2 <- matrix(0, n, 2)
    storage.mode(datatyp2) <- "double"
    pxpy <- matrix(0, n, 3)
```

```
    storage.mode(pxpy) <- "double"
    whisk <- 2
    alpha <- alpha*100
    bpvals <- .Fortran("abagplot", as.integer(n),
                       as.integer(alpha), x, y, as.integer(
                           whisk),
                       tukm = double(2), interpx = interpx,
                           interpy = interpy,
                       num = as.integer(0), datatyp =
                           datatyp, indoutl = integer(n),
                       datatyp2 = datatyp2, pxpy = pxpy,
                           boxpl = as.integer(0),
                       nointer = as.integer(0))
    bpx <- bpvals$interpx
    bpy <- bpvals$interpy
    bpxynn <- (bpx != 0) & (bpy != 0)
    if(any(bpxynn)) {
        return(data.frame(x = bpx[bpxynn],y= bpy[bpxynn]))
    } else {
        return(NULL)
    }
}
```

Listing B.16: `mltv::createconcellipse` function

```
##' Function to create list of input for concentration ellipses.
##'
##' @param mltv : Object of class \code{mltv}.
##' @param alpha : alpha value for \code{concellipse} function.
##'
##' @return List with groupnames in \code{mltv$groups} containing
    coordinates for
##' concentration ellipses for the groups.  The list is added to
    the "container" slot
##' in the \code{mltv} object.
##'
##' @export

createconcellipse <- function(mltv,alpha=0.90,...) {
    assert_that(inherits(mltv,"mltv"))
    ce <- list()
    for(grp in unique(mltv$groups)) {
        ce[[grp]] <- concellipse(mltv$projpoints[mltv$groups==grp
            ,1],
                                 mltv$projpoints[mltv$groups==grp,2],
                                    alpha=alpha,...)
    }
    mltv[["container"]] <- ce
    return(mltv)
}
```

Listing B.17: `mltv::concellipse` function

```
##' Function to return concentration ellipse points for plotting
##' It does not plot the ellipse.
##'
##' @param x : x values
##' @param y : y values
##' @param alpha : alpha value for the ellipse
##'
##' @export

concellipse <- function(x,y,alpha=0.9,...) {
  covmat <- cov(cbind(x,y))
  concellipse <- ellipse(covmat,level=alpha,centre=c(mean(x),mean(
      y)),...)
  colnames(concellipse) <- c("x","y")
  return(as.data.frame(concellipse))
}
```

Listing B.18: `mltv::createconvexhull` function

```
##' Function to create list of input for convexhulls.
##'
##' @param mltv : Object of class \code{mltv}.
##' @param layer : The layer to return. For example, 1 is the
    outer layer. 2 is the convex hulle with outer layer removed etc
    .
##'
##' @return list with groupnames in \code{mltv$groups} containing
    coordinates for convexhulls
##' for the groups. The list is added to the "container" slot in
    the \code{mltv} object.
##'
##' @export

createconvexhull <- function(mltv,layer=1) {

    assert_that(inherits(mltv,"mltv"))

    cv <- list()
    for(grp in unique(mltv$groups)) {
        cv[[grp]] <- convexhull(mltv$projpoints[mltv$groups==grp
            ,1],
                                mltv$projpoints[mltv$groups==grp,2],
                                    layer = layer)
    }
    mltv[["container"]] <- cv
    return(mltv)

}
```

Listing B.19: `mltv::convexhull` function

```
##' Function to return convex hull points for plotting
##' It does not plot the hull.
##'
##' @param x : x values
##' @param y : y values
##' @param layer : The layer to return. For example, 1 is the
    outer layer. 2 is the convex hulle with outer layer removed etc
    .
##'
##' @export

convexhull <- function(x,y,layer=1) {
    xmat <- cbind(x,y)
    for(i in 1:layer) {
        convexhull <- chull(xmat)
        if(i < layer)
            xmat <- xmat[-convexhull,]

    }
    convexhull <- xmat[convexhull,]
    colnames(convexhull) <- c("x","y")
    return(as.data.frame(convexhull))
}
```

B.1.5 Axis label and marker calculations

Listing B.20: `mltv::calcgowaxes` function

```
##' Function to create object with all the information needed to draw the Gower biplot axes
##'
##' @param mltv Object of one of the \code{mltv} classes.
##'
##'
##' @return gowaxes object containing:
##' \itemize{
##' \item \code{axcoef} - The ax coefficient values.
##' \item \code{axes} - for each of the axis:
##' \itemize{
##' \item \code{slope} - The slope of the axes.
##' \item \code{side} - Which side of the plot the axis label should be.
##' \item \code{at} - Where on the side of the plot should the axis label be.
##' \item \code{markangle} - The angle of the tick marks on the axis.
##' \item \code{markerpos} - The position of the tick marks, and labels for the axis.
##' \item \code{markers} - The numeric labels for the tick marks.
##' }
##' }
##' @export

calcgowaxes <- function(mltv) {
    gowaxes <- list()
    axnames <- mltv$axnames
    extvals <- mltv$extvals
    Vr <- mltv$vraxes

    axcoef <- solve(diag(diag(Vr%*%t(Vr)))) %*% Vr
    gowaxes[["axcoef"]] <- axcoef
    for(i in mltv$axes) {
        ##Calculates slope
        m <- Vr[i,2]/Vr[i,1]
        ##y at xmin > ymin and y at xmin is < ymax and ax direction is negative on x
        gowaxes[["axes"]][[axnames[i]]] <- list()
        gowaxes[["axes"]][[axnames[i]]][["slope"]] <- m
        if(extvals[3]<= m*extvals[1] && m*extvals[1]<=extvals[4]&& Vr[i,1]< 0) {
            gowaxes[["axes"]][[axnames[i]]][["side"]] <- 2
            gowaxes[["axes"]][[axnames[i]]][["at"]] <- m*extvals[1]
        }
        ##y at xmin is > ymax and ax direction is positive on y
        ##or y at xmax is > ymax and  ax direction is positive on y
```

```
41          if((m*extvals[1]>extvals[4]&&Vr[i,2]>0)||(m*extvals[2]>
                extvals[4]&&Vr[i,2]>0)) {
42              gowaxes[["axes"]][[axnames[i]]][["side"]] <- 3
43              gowaxes[["axes"]][[axnames[i]]][["at"]] <- extvals[4]/
                    m
44
45          }
46          if((m*extvals[1]<extvals[3]&&Vr[i,2]<0)||(m*extvals[2]<
                extvals[3]&&Vr[i,2]<0)) {
47              gowaxes[["axes"]][[axnames[i]]][["side"]] <- 1
48              gowaxes[["axes"]][[axnames[i]]][["at"]] <- extvals[3]/
                    m
49
50          }
51          if(extvals[3]<= m*extvals[2]&&m*extvals[2]<=extvals[4]&&Vr
                [i,1]>0) {
52              gowaxes[["axes"]][[axnames[i]]][["side"]] <- 4
53              gowaxes[["axes"]][[axnames[i]]][["at"]] <- m*extvals
                    [2]
54
55          }
56
57
58          gowaxes[["axes"]][[axnames[i]]][["markangle"]] <- 180*atan
                (-1/m)/pi
59          Xmarkers <- axmarkerseq(mltv,i,5,mltv$scaled)
60          Xmarkers <- pretty(Xmarkers,5,min.n = 5)
61
62          Xcentmarkers <- Xmarkers - mltv$colmeans[i]
63          markers <- cbind(Xcentmarkers*axcoef[i,1],Xcentmarkers*
                axcoef[i,2])
64          gowaxes[["axes"]][[axnames[i]]][["markerpos"]] <- markers
65          gowaxes[["axes"]][[axnames[i]]][["markers"]] <- Xmarkers
66
67
68      }
69      mltv[["gowaxes"]] <- gowaxes
70      return(mltv)
71  }
```

Listing B.21: `mltv::axmarkerseq` function

```
##' Function to calculate tick marks for biplot axes
##'
##' @param mltv Object of one of the \code{mltv} classes.
##' @param ind Numeric value indicating which axis to calculate markers for.
##' @param n.int Number of markers to calc.
##' @param scl Are the data scaled.
##'
##' @return Numeric vector of \code{n.int} values between the upper and lower limits
##' for the \code{ind}-th biplot axis.
##'
##' @export


axmarkerseq <- function(mltv,ind,n.int,scl=FALSE){
  a <- mltv$extvals
  V <- mltv$vraxes[ind,]
  m <- V[2]/V[1]
  rng <- matrix(0,2,2)

  if(m>=0) {
    if(a[2]*m > a[4]) {
        rng[1,2] <- a[4]
        rng[1,1] <- a[4]/m
    } else {
        rng[1,1] <- a[2]
        rng[1,2] <- a[2]*m
    }
    if(a[1]*m < a[3]){
        rng[2,1] <- a[3]/m
        rng[2,2] <- a[3]
    } else {
        rng[2,2] <- a[1]*m
        rng[2,1] <- a[1]
    }

  } else {
      if(a[1]*m > a[4]) {
          rng[1,2] <- a[4]
          rng[1,1] <- a[4]/m
      } else {
          rng[1,1] <- a[1]
          rng[1,2] <- a[1]*m
      }

    if(a[2]*m < a[3]) {
        rng[2,1] <- a[3]/m
        rng[2,2] <- a[3]
    } else {
        rng[2,2] <- a[2]*m
```

```
50            rng[2,1] <- a[2]
51
52        }
53    }
54    rngunsc <- rng%*%t(mltv$vraxes)
55
56    meanX.mat <- matrix(rep(mltv$colmeans,nrow(rngunsc)),nrow=nrow(
          rngunsc),byrow=TRUE)
57
58    sdX.mat <- matrix(rep(mltv$colsd,nrow(rngunsc)),nrow=nrow(
          rngunsc),byrow=TRUE)
59    if(scl){
60        rngunscun <- (rngunsc*sdX.mat)+meanX.mat
61    } else {
62        rngunscun <- rngunsc + meanX.mat
63    }
64    rng <- c(min(rngunscun[,ind]),max(rngunscun[,ind]))
65    if((rng[2]-rng[1])>10){
66        rnd <- 0
67    } else {
68        rnd <- 2
69    }
70    return(round(seq(rng[1],rng[2],length.out=n.int+2)[2:(n.int+1)],
          rnd))
71
72 }
```

B.1.6 Biplot theme functions

Listing B.22: `mltv::createmltvtheme` function

```
##' Function to create generic theme
##'
##' @param mltv Object of one of the \code{mltv} classes.
##'
##' @export
##'



createmltvtheme <- function(mltv) {

    assert_that(inherits(mltv,"mltv"))

    mltvtheme <- list()
    mltvtheme[["fill"]] <- "gray80"
    mltvtheme[["titlecol"]] <- "gray30"
    mltvtheme[["axes"]] <- list()
    mltvtheme[["points"]] <- list()
    mltvtheme[["container"]] <- list()
    mltvtheme[["groups"]] <- list()
    mltvtheme$axes[["labels"]] <- list()
    mltvtheme$axes$labels[["padding"]] <- unit(5,"mm")
    mltvtheme$axes[["colfunc"]] <- biplotguicolors
    mltvtheme$groups[["colfunc"]] <- biplotguicolors
    mltvtheme$groups[["pch"]] <- 19
    mltvtheme$container[["alpha"]] <- 0.20
    mltvtheme$points[["pchfunc"]] <- pchfunc
    mltvtheme$points[["alpha"]] <- 0.2
    mltvtheme$plot$axes <- TRUE
    mltvtheme$plot$container <- ("container" %in% names(mltv))
    mltvtheme$plot$means <- inherits(mltv,"mltv_cva")
    mltvtheme$plot$points <- (!("xnew" %in% names(mltv)))&(!
        inherits(mltv,"mltv_cva"))
    mltvtheme$plot$newpoints <- ("xnew" %in% names(mltv))
    mltvtheme$xpadding <- c(1.15,1.15)
    mltvtheme$ypadding <- c(1.15,1.15)
    class(mltvtheme) <- c("mltv_theme")
    return(mltvtheme)

}
```

Listing B.23: `mltv::biplotguicolors` function

```
#' Function that returns BiplotGUI colours
#'
#' Function to generate colours like biplotGUI defaults
#'
#' @param n : number of colours to generate
#'
#' @export
#'
#'
biplotguicolors <- function(n) {
  biplotguicolors <- hcl(seq(0, 360, length = n +
                    2)[-c(1, n + 2)], l=40, c=110)
  return(biplotguicolors)
}
```

Listing B.24: `mltv::ggplotcolors` function

```
#' Function that returns GGplot colours
#'
#' Function to generate colours like ggplot defaults
#'
#' @param n : number of colours to generate
#'
#' @export
#'
#'
ggplotcolors <- function(n) {
  hues = seq(15, 375, length=n+1)
  ggplotcolorhue <- hcl(h=hues, l=65, c=100)[1:n]
  return(ggplotcolorhue)
}
```

Listing B.25: `mltv::pchfunc` function

```
##' Function that returns n pch values
##'
##' @param n : Number of pch values to return
##'
##' @return n pch values
##'
##' @export
##'

pchfunc <- function(n) {
    pchs <- rep(21:25,ceiling(n/4))
    pchs <- pchs[1:n]
    return(pchs)

}
```

B.1.7 Biplot utility functions

Listing B.26: `mltv::calcextvals` function

```
##' Function to calculate extreme values of plot from data and
    theme settings.
##'
##' @param mltv Object of one of the \code{mtlv} classes.
##' @param mltvtheme Output from \code{createmltvthem}.
##'
##' @return vector of extreme values c(xmin,xmax,ymin,ymax) in \
    code{native} units
##' inserted in "extvals" slot of \code{mltv} object.
##'
##'
##' @export
##'



calcextvals <- function(mltv,mltvtheme) {

    assert_that(inherits(mltv,"mltv"))

    xmin <- xmax <- ymin <- ymax <- NA
    xmin <- min(xmin,mltv$projpoints[,1],na.rm=TRUE)
    xmax <- max(xmax,mltv$projpoints[,1],na.rm=TRUE)
    ymin <- min(ymin,mltv$projpoints[,2],na.rm=TRUE)
    ymax <- max(ymax,mltv$projpoints[,2],na.rm=TRUE)

    if(mltvtheme$plot$means) {
        xmin <- min(xmin,mltv$projmean[,1],na.rm=TRUE)
        xmax <- max(xmax,mltv$projmean[,1],na.rm=TRUE)
        ymin <- min(ymin,mltv$projmean[,2],na.rm=TRUE)
        ymax <- max(ymax,mltv$projmean[,2],na.rm=TRUE)
    }
    if(mltvtheme$plot$container) {
        for(grp in names(mltv$container)) {
            xmin <- min(xmin,mltv$container[[grp]][,1],na.rm=TRUE)
            xmax <- max(xmax,mltv$container[[grp]][,1],na.rm=TRUE)
            ymin <- min(ymin,mltv$container[[grp]][,2],na.rm=TRUE)
            ymax <- max(ymax,mltv$container[[grp]][,2],na.rm=TRUE)
        }
    }
    if(mltvtheme$plot$newpoints) {
        xmin <- min(xmin,mltv$xnew$xnewprojectedpoints[,1],na.rm=
            TRUE)
        xmax <- max(xmax,mltv$xnew$xnewprojectedpoints[,1],na.rm=
            TRUE)
        ymin <- min(ymin,mltv$xnew$xnewprojectedpoints[,2],na.rm=
            TRUE)
        ymax <- max(ymax,mltv$xnew$xnewprojectedpoints[,2],na.rm=
            TRUE)
```

```
    }

    extvals <- c(xmin,xmax,ymin,ymax)

    assert_that(!any(is.na(extvals)))

    if(!is.null(mltvtheme$xpadding)) {
                    extvals[1:2] <- extvals[1:2]*mltvtheme$
                        xpadding
    }
    if(!is.null(mltvtheme$ypadding)) {
        extvals[3:4] <- extvals[3:4]*mltvtheme$xpadding
    }

    mltv[["extvals"]] <- extvals
    return(mltv)
}
```

Listing B.27: `mltv::gmembermat` function

```
##' Function to calculate group membership function
##'
##' Function to create matrix of group membership with G[i,j]==1
    if sample i belongs to group j, and 0 otherwise
##'
##' @param g : vector of length n with group membership for each
    sample
##'
##' @return G : matrix of group membership with G[i,j]==1 if
    sample i belongs to group j, and 0 otherwise
##'
##' @export

gmembermat <- function (g)
{
    groups <- unique(g)
    n <- length(g)
    p <- length(groups)
    G <- matrix(0, nrow = n, ncol = p)
    colnames(G) <- groups
    for (i in 1:p) {
      G[g == groups[i],i] <- 1
    }
    return(G)
}
```

Listing B.28: `mltv::maxstring` function

```
##' Function that returns maximum string length
##'
##' Wrapper function to return the maximum string grob length.
##' Initial use case is to get a unit to use as width and height of margins
##' in biplot package. This will ensure that all the sides is equal, and just
##' big enough to hold the strings.
##'
##' @param labels Vector of strings
##'
##' @return Grid unit object returned from max function
##'
##' @export
##'


maxstring <- function(labels) {
    mxstr <- max(unit.c(stringWidth(labels)))
    return(mxstr)
}
```

Listing B.29: `mltv::getaxlabels` function

```
##' Function to return ax labels in a convenient format.
##'
##' Change the format of the ax labels to a more convenient format for including
##' in the plotting function. Will be redundant if grid functions are refactored.
##' Note that both the \code{calcgowaxes}, and \code{calcextvals} should precede
##' this function.
##'
##' @param mltv Object of one of the \code{mltv} classes.
##'
##' @return List with four lists (one for each side of the plot), included in the
##' "axlabels" slot of the \code{mltv} object.
##'
##' @export

getaxlabels <- function(mltv) {

    assert_that(inherits(mltv,"mltv"))

    axlabels <- list(side1=list(),side2=list(),side3=list(),side4=list())
    for(ax in names(mltv$gowaxes$axes)) {

        if(mltv$gowaxes$axes[[ax]]$side==1) {
```

```
23              axlabels$side1[[ax]] <- (mltv$gowaxes$axes[[ax]]$at-
                    mltv$extvals[1])/
24                  (mltv$extvals[2]-mltv$extvals[1])
25          }
26          if(mltv$gowaxes$axes[[ax]]$side==2) {
27              axlabels$side2[[ax]] <- (mltv$gowaxes$axes[[ax]]$at-
                    mltv$extvals[3])/
28                  (mltv$extvals[4]-mltv$extvals[3])
29          }
30          if(mltv$gowaxes$axes[[ax]]$side==3) {
31              axlabels$side3[[ax]] <- (mltv$gowaxes$axes[[ax]]$at-
                    mltv$extvals[1])/
32                  (mltv$extvals[2]-mltv$extvals[1])
33          }
34          if(mltv$gowaxes$axes[[ax]]$side==4) {
35              axlabels$side4[[ax]] <- (mltv$gowaxes$axes[[ax]]$at-
                    mltv$extvals[3])/
36                  (mltv$extvals[4]-mltv$extvals[3])
37          }
38      }
39      mltv[["axlabels"]] <- axlabels
40      return(mltv)
41  }
```

B.1.8 Biplot plotting functions

Listing B.30: `mltv::mltvbipl` function

```
##' Generic method to draw multivariate grid plot
##'
##' @param mltv Object of one of the \code{mtlv} classes.
##' @param mltvtheme Output from \code{createmltvthem}.
##' @param main Title for the plot
##'
##' @export
##'


mltvbipl <- function(mltv,mltvtheme,main) {
    UseMethod("mltvbipl")
}
```

Listing B.31: `mltv::mltvbipl.mltv` function

```
##' Main function to draw multivariate grid plot
##'
##' @param mltv Object of one of the \code{mtlv} classes.
##' @param mltvtheme Output from \code{createmltvthem}.
##' @param main Title for the plot
##'
##' @return A list with two objects:
##' \itemize{
##' \item \code{mltv} - The original \code{mltv} object updated
    with internal
##' plot calculations.
##' \item \code{mltvtheme} - The original \code{mltvtheme} object.
##' }
##' @export
##'


mltvbipl.mltv <- function(mltv,mltvtheme,main=NULL) {


    mltv <- calcextvals(mltv,mltvtheme)


    if("axispred" %in% names(mltv)) {
        mltv$axes <- axisinclude(mltv$axispred)
        mltv$axnames <- paste(mltv$axnames," (",round(mltv$
            axispred$axispred,2),")",sep="")
    }

    mltv <- calcgowaxes(mltv)

    grid.newpage()
    mxs <- maxstring((mltv$axnames))
```

```
32
33      vp <- viewport(layout=grid.layout(4,3,widths=unit.c(mxs+
            mltvtheme$axes$labels$padding,
34                                                          unit(1,"null"),mxs+
                                                                mltvtheme$axes$
                                                                labels$padding),
35                             heights=unit.c(unit(3,"line"),mxs+mltvtheme
                                   $axes$labels$padding,
36                                    unit(1,"null"),mxs+mltvtheme$axes$
                                        labels$padding),
37                             respect=matrix(c(rep(0,7),1,rep(0,4)),ncol
                                   =3)),gp=gpar(cex=0.75))
38      grid.rect(gp=gpar(fill=mltvtheme$fill))
39
40      pushViewport(vp)
41
42      pushViewport(dataViewport(xscale=mltv$extvals[1:2],yscale =
            mltv$extvals[3:4],extension = 0,
43                                layout.pos.row=3,layout.pos.col=2,
                                      clip=TRUE))
44
45
46      axcols <- mltvtheme$axes$colfunc(length(mltv$axnames))
47      names(axcols) <- mltv$axnames
48
49      for(ax in names(mltv$gowaxes$axes)) {
50           grid.abline(intercept=0,slope=mltv$gowaxes$axes[[ax]]$
                slope,
51                       gp=gpar(col=as.vector(axcols[ax])))
52           grid.points(mltv$gowaxes$axes[[ax]]$markerpos[,1],
53                       mltv$gowaxes$axes[[ax]]$markerpos[,2],
54                       pch=mltvtheme$groups$pch,
55                       gp=gpar(col=mltvtheme$fill,fill=mltvtheme$fill
                             ))
56           grid.text(mltv$gowaxes$axes[[ax]]$markers,mltv$gowaxes$
                axes[[ax]]$markerpos[,1],
57                     mltv$gowaxes$axes[[ax]]$markerpos[,2],
58                     gp=gpar(cex=0.75,col=as.vector(axcols[ax])),
                          default.units = "native",
59                     rot=mltv$gowaxes$axes[[ax]]$markangle)
60      }
61      mltv <- getaxlabels(mltv)
62      curlineheight <- as.numeric(convertX(unit(1,"line"),"npc"))
63
64      ##mltv$axlabels <- shakeaxelabels(mltv$axlabels,curlineheight
            +0.2)
65      grpcolors <- mltvtheme$groups$colfunc(length(unique(mltv$
            groups)))
66      grppch <- mltvtheme$points$pchfunc(length(unique(mltv$groups))
            )
67      names(grpcolors) <- unique(mltv$groups)
68      names(grppch) <- unique(mltv$groups)
```

```
69      if(!is.null(mltv$container) & mltvtheme$plot$container) {
70          for(grp in names(grpcolors))
71              grid.polygon(mltv$container[[grp]]$x,mltv$container[[
                    grp]]$y,
72                           gp=gpar(col=as.vector(grpcolors[grp]),
                                 alpha=mltvtheme$container$alpha,
73                                  fill=as.vector(grpcolors[grp])),
                                     default.units="native")
74      }
75      if(mltvtheme$plot$points) {
76          for(grp in unique(mltv$groups)) {
77              grid.points(mltv$projpoints[mltv$groups==grp,1],mltv$
                    projpoints[mltv$groups==grp,2],
78                          gp=gpar(col=as.vector(grpcolors[grp]),
                                alpha=mltvtheme$points$alpha,fill=as.
                                vector(grpcolors[grp])),default.units =
                                 "native",pch = as.vector(grppch[grp])
79                          )
80
81          }
82
83      }
84      if(mltvtheme$plot$newpoints) {
85         ## for(grp in unique(mltv$groups)) {
86          ##    grid.points(mltv$xnew$xnewprojectedpoints[,1],
87          ##                mltv$xnew$xnewprojectedpoints[,2], gp=gpar(
                col="black",alpha=mltvtheme$points$alpha,fill="black"),
                default.units = "native",pch = 20)
88          grid.text(rownames(mltv$xnew$projpoints),mltv$xnew$
                projpoints[,1],
89                    mltv$xnew$projpoints[,2],default.units = "
                       native")
90
91          ##}
92
93      }
94      if(mltvtheme$plot$means) {
95          grid.text(rownames(mltv$projmean),mltv$projmean[,1],mltv$
                projmean[,2],
96                    gp=gpar(col=as.vector(grpcolors),fontface="bold"
                        ),default.units = "native")
97      }
98      grid.rect()
99      popViewport()
100     pushViewport(viewport(layout.pos.row=3,layout.pos.col=1,name="
            Side2"))
101     if(length(mltv$axlabels$side2)>0) {
102         for(ax in names(mltv$axlabels$side2))
103             grid.text(ax,x=unit(1,"npc")-unit(1,"mm"),y=unit(mltv$
                    axlabels$side2[[ax]],"npc"),just="right",
104                       gp=gpar(col=as.vector(axcols[ax]),fontface="
                           bold"))
```

```
105
106     }
107     popViewport()
108     pushViewport(viewport(layout.pos.row=2,layout.pos.col=2,name="
            Side3"))
109     if(length(mltv$axlabels$side3)>0) {
110         for(ax in names(mltv$axlabels$side3)) {
111
112             grid.text(ax,x=unit(mltv$axlabels$side3[[ax]]-as.
                    numeric(convertX(stringHeight(ax),"npc"))/2,"npc"),
113                     y=unit(as.numeric(convertY(stringWidth(ax),"
                            native"))/2,"npc")+unit(1,"mm"),
114                     just=c("centre","top"),rot=90,gp=gpar(col=as
                            .vector(axcols[ax]),fontface="bold"))
115         }
116
117     }
118
119     popViewport()
120     pushViewport(viewport(layout.pos.row=3,layout.pos.col=3,name="
            Side4"))
121     if(length(mltv$axlabels$side4)>0) {
122         for(ax in names(mltv$axlabels$side4))
123             grid.text(ax,x=unit(0,"npc")+unit(1,"mm"),y=unit(mltv$
                    axlabels$side4[[ax]],"npc"),just="left",
124                     gp=gpar(col=as.vector(axcols[ax]),fontface="
                            bold"))
125
126     }
127     popViewport()
128     pushViewport(viewport(layout.pos.row=4,layout.pos.col=2,name="
            Side1"))
129     if(length(mltv$axlabels$side1)>0) {
130         for(ax in names(mltv$axlabels$side1)) {
131             grid.text(ax,x=unit(mltv$axlabels$side1[[ax]]+as.
                    numeric(convertX(stringHeight(ax),"npc"))/2,"npc"),
132                     y=unit(1-as.numeric(convertY(stringWidth(ax)
                            ,"native"))/2,"npc")-unit(1,"mm"),
133                     just=c("centre","bottom"),rot=90,gp=gpar(col
                            =as.vector(axcols[ax]),fontface="bold"))
134         }
135
136     }
137     popViewport()
138
139     pushViewport(viewport(layout.pos.row=1,layout.pos.col=1:3))
140     if(is.null(main)) {
141         main <- paste("Plot of PC",mltv$evec[1]," vs PC",mltv$evec
                [2],
142                     " (",round(mltv$quality,1),"% total info)",
                        sep="")
143     } else {
```

```
144             main <- paste("Plot of PC",mltv$evec[1]," vs PC",mltv$evec
                    [2],
145                           " (",round(mltv$quality,1),"% total info) -
                              ",main,sep="")
146         }
147         grid.text(main,x=unit(0.5,"npc"),y=unit(0.5,"npc"),just="
                centre",
148                   gp=gpar(col=mltvtheme$titlecol,fontface="bold",
                        fontsize=16))
149         popViewport()
150         return(list(mltv=mltv,mltvtheme=mltvtheme))
151     }
```

B.2 Code Listings for GOPA

Listing B.32: `GOPA` functions

```
1
2  ##' This file provides a function to perform the GOPA analysis, as
       well as functions to create a PCA biplot of the GOPA output.
3  ##' To create the CVA and monitoring biplots (as well as various
       other biplots) the reader are
4  ##' advised to use the UBbipl R package provided and documented in
       the book Understanding Biplots:
5  ##'
6  ##' Gower, J., Lubbe, S., Le Roux, N., 2011. Understanding Biplots
       . Chichester:
7  ##' John Wiley & Sons.
8  ##'
9  ##' Dropbox links to the newest versions of the UBbipl package are
       provided:
10 ##' https://dl.dropboxusercontent.com/u/17860902/UBbipl_3.0.4.tar.
       gz
11 ##' https://dl.dropboxusercontent.com/u/17860902/UBbipl_3.0.4.zip
12 ##'
13 ##'
14 ##'
15 ##' ##' Generalized Orthogonal Procrustes Analysis
16 ##'
17 ##' Function to perform GOPA. This function was originally
       published as a suplement to
18 ##' Arnold, G.M., Gower, J.C., Gardner-Lubbe, S., le Roux, N.J.,
       2007. Biplots
19 ##' of free-choice profile data in Generalized Orthogonal
       Procrustes Analyis.
20 ##' Applied Statistics 56, 445-458.
21 ##' The function was updated to be more general in application.
22 ##'
23 ##' @param Xk List of matrices for GOPA analysis.
24 ##' @param K Number of matrices in Xk.
25 ##' @param pk Vector of length K consisting ncol for each matrix
       in Xk.
26 ##' @param istoropic Should isotropic scaling be performed.
27 ##' @param Pk.scaling Should pk-scaling be performed.
28 ##' @param eps Tolerance paramameter for GOPA algorithm
29 ##'
30 GOPA <- function(Xk, K, pk, isotropic, Pk.scaling, eps)
31 {
32     ## Perform Orthogonal Procrustes Analysis
33     OPA <- function(X.mat, Z.mat) {
34         svd.zx <- svd(t(Z.mat) %*% X.mat)
35         svd.zx$v %*% t(svd.zx$u)
36     }
37     ##
38     ## Method: The Xk matrices updated with each iteration
```

```
    ##
    ## Preparation
    ##
    n <- nrow(Xk[[1]])
    p <- max(pk)
    means <- sapply(Xk, function(X)
                    apply(X, 2, mean))
    if(Pk.scaling){
        Xk.scale <- sapply(Xk, function(X)
                           scale(X)/(sqrt(ncol(X)) * sqrt(nrow(X)
                              - 1)),simplify=F)
    } else {
        Xk.scale <- sapply(Xk, function(X) scale(X, center = T,
            scale = rep(sqrt(sum(eigen(t(scale(X, center = T, scale
             = F)) %*% scale(X, center = T, scale = F))$values)),
            ncol(X))),simplify=F)
    }

    p <- max(pk)
    for(i in 1:K) {
        if(pk[i]<p)
            Xk.scale[[i]] <- cbind(Xk.scale[[i]], matrix(0, nrow =
                                                             nrow(Xk.
                                                                  scale
                                                                  [[i]])
                                                             , ncol
                                                              = p -
                                                              pk[i
                                                             ]))
        Xk.scale[[i]][is.nan(Xk.scale[[i]])] <- 0
    }
    Qk <- sapply(1:K, function(k, p)
                 return(diag(p)), p = max(pk), simplify = F)
    sk <- rep(1, K)
    sk.F <- sk
    Xk.F <- sapply(1:K, function(k, Xk, sk, Qk)
                   sk[k] * Xk[[k]] %*% Qk[[k]], Xk = Xk.scale, sk
                       = sk, Qk = Qk,
                   simplify = F)
    Qk.F <- sapply(1:K, function(k, Qk)
                   Qk[[k]], Qk = Qk, simplify = F)
    tel <- 0
    sum.sq.old <- Inf
    repeat
        {tel <- tel + 1
         if(tel > iter)stop("Maximum number of specified
            iterations reached! Increase iter \n")
         Xk.F <- sapply(1:K, function(k, Xk, sk, Qk)
                        sk[k] * Xk[[k]] %*% Qk[[k]], Xk = Xk.F, sk
                            = sk, Qk =
                    Qk, simplify = F)
         Gmat <- Xk.F[[1]]
```

```
78          for(k in 2:K)
79              Gmat <- Gmat + Xk.F[[k]]
80          Gmat <- Gmat/K
81          Qk <- sapply(1:K, function(k, Xk, Gmat)
82                       OPA(Xk[[k]], Gmat), Xk = Xk.F, Gmat = Gmat,
                             simplify =
83                       F)
84          Qk.F <- sapply(1:K, function(k, Qk, QF)
85                         QF[[k]] %*% Qk[[k]], Qk = Qk, QF = Qk.F,
                               simplify = F)
86          if(isotropic) {
87              Smat <- matrix(0, ncol = K, nrow = K)
88              for(i in 1:K)
89                  for(j in i:K) {
90                      Smat[i, j] <- sum(diag(t(Qk[[i]]) %*% t(
91                          Xk.F[[i]]) %*% Xk.F[[j]] %*% Qk[[j]]))
92                      if(i != j)
93                          Smat[j, i] <- Smat[i, j]
94                  }
95              Smat.min.half <- diag(1/sqrt(diag(Smat)))
96              swd <- svd(Smat.min.half %*% Smat %*%
97                         Smat.min.half)
98              sk <- Smat.min.half %*% swd$u[, 1] * sqrt(K)
99              if(sk[1] < 0)
100                 sk <- -1 * sk
101         }
102         else {
103             Smat <- matrix(0, ncol = K, nrow = K)
104             for(i in 1:K)
105                 for(j in i:K) {
106                     Smat[i, j] <- sum(diag(t(Xk.F[[i]]) %*%
107                                            Xk.F[[j]]))
108                     if(i != j)
109                         Smat[j, i] <- Smat[i, j]
110                 }
111             Smat.min.half <- diag(1/sqrt(diag(Smat)))
112             swd <- svd(Smat.min.half %*% Smat %*%
113                        Smat.min.half)
114             sk <- sk
115         }
116         sk.F <- sk.F * sk
117         sum.sq <- sum(sapply(1:K, function(k, Xk, Gmat)
118                              sum(diag((Xk[[k]] - Gmat) %*% t(Xk[[
                                     k]] - Gmat))), Xk
119                              = Xk.F, Gmat = Gmat))
120         cat("iter", tel, "sum.sq: ", sum.sq, "\n")
121         if((sum.sq.old - sum.sq) < eps) break
122         sum.sq.old <- sum.sq
123     }
124     sum.sq.all <- sapply(1:K, function(k, Xk, Gmat)
125                              sum(diag((Xk[[k]] - Gmat) %*% t(Xk[[
                                     k]] - Gmat))), Xk
```

```
126                                     = Xk.F, Gmat = Gmat)
127
128     return(list(sk=sk, sk.F=sk.F, Xk.scale=Xk.scale, Gmat=Gmat, Qk
            =Qk, Qk.F=Qk.F, Xk.F=Xk.F, p=p, n=n, means=means,sum.sq=sum
            .sq,sum.sq.all=sum.sq.all))
129 }
130
131 ##' This function is a utility function to draw the biplot axes
132 ##'
133
134 DrawBiplotAxes <- function(V.mat,names.vek = paste("V",1:nrow(V.
        mat)),
135                                     CeX=0.6,kk=.1,cols=rep("black",nrow(V.
                                            mat)))
136 {
137     a <- par("usr")
138
139     for(i in 1:nrow(V.mat))
140         {
141             if(V.mat[i,1] != 0) {
142             m <- V.mat[i,2]/V.mat[i,1]
143             abline(a=0,b=m, col=cols[i])
144             if(a[3]<= m*a[1] && m*a[1]<=a[4]&& V.mat[i,1]< 0)
145                 {
146                     mtext(names.vek[i],side=2,at=m*a[1],las=2,cex
                            = CeX,adj=12*kk,col=cols[i])
147                 }
148
149             if(m*a[1]>a[4]||m*a[2]>a[4])
150                 {
151                     if(V.mat[i,2] > 0)
152                         {
153                             mtext(names.vek[i],side=3,at=a[4]/m,
                                    las=2,cex = CeX,adj=-kk,col=cols[i
                                    ])
154                         }
155                 }
156
157             if(m*a[1]<a[3]||m*a[2]<a[3])
158                 {
159                     if(V.mat[i,2] < 0)
160                         {
161                             mtext(names.vek[i],side=1,at=a[3]/m,
                                    las=2,cex = CeX,adj=1,col=cols[i])
162                         }
163                 }
164
165             if(a[3]<= m*a[2]&&m*a[2]<=a[4]&&V.mat[i,1]>0)
166                 {
167                     mtext(names.vek[i],side=4,at=m*a[2],las=2,cex=
                            CeX,adj=-kk,col=cols[i])
168                 }
```

```
169             }
170
171             }
172
173 }
174
175 ##' Function to create PCA biplots of the GOPA output
176 ##'
177 ##' This function can be used to replicate the PCA biplots from
        the GOPA output discussed in the paper.
178 ##'
179 ##' @param GOPA.out The output from the GOPA function
180 ##' @param axes Should biplot axes be added to the plot
181 ##' @param legpos Where should the legend be positioned. This
        parameter are sent \
182 ##' directly to the \code{legend} function in R, and should
        therefore be of the same format.
183
184
185 GOPAplot <- function(GOPA.out, axes=FALSE, legpos="topright",
        nraxcols=NULL) {
186
187     Gmat <- GOPA.out$Gmat
188     if(is.null(colnames(Gmat))) {
189         colnames(Gmat) <- paste("V",1:ncol(Gmat),sep="")
190     }
191     Gmatsvd <- svd(t(Gmat)%*%Gmat)
192
193     V <- Gmatsvd$u[,1:2]
194     Gmatr <- Gmat%*%V
195     Xk.Fr <- NULL
196     Xk.F <- GOPA.out$Xk.F
197
198     cols <- hcl(seq(0, 360, length = nrow(Gmatr) +
199                     2)[-c(1, nrow(Gmatr) + 2)], l=40, c=110)
200     if(is.null(names(Xk.F))) {
201         names(Xk.F) <- 1:length(Xk.F)
202     }
203     for(i in names(Xk.F)) {
204         tmp <- cbind(Xk.F[[i]]%*%V,rownames(Xk.F[[i]]),i)
205         colnames(tmp) <- c("x","y","grp","K")
206         Xk.Fr <- rbind(Xk.Fr,tmp)
207     }
208     plot(Gmatr,type="n",asp=1,ann=F,axes=F)
209     txtcols <- rep(cols,length(Xk.F))
210     text(Xk.Fr[,c("x","y")],labels=Xk.Fr[,"K"],col=txtcols,cex
            =0.65)
211     box()
212     for(i in 1:nrow(Gmat)) {
213         tmp <- Xk.Fr[Xk.Fr[,"grp"]==rownames(Gmat)[i],]
214         tmphull <- chull(tmp[,c("x","y")])
215         polygon(tmp[tmphull,c("x","y")],density=25,col=cols[i])
```

```
216     }
217     if(axes) {
218         if(is.null(nraxcols)) {
219         axcols <- hcl(seq(0, 360, length = nrow(V) +
220                                 2)[-c(1, nrow(V) + 2)], l=40, c=110)
221     } else {
222         axcols <- hcl(seq(0, 360, length = nraxcols +
223                                 2)[-c(1, nraxcols + 2)], l=40, c
                                    =110)
224         axcols <- rep(axcols,nrow(V)/nraxcols)
225     }
226         DrawBiplotAxes(V,colnames(Gmat),cols=axcols)
227         for(i in 1:ncol(Gmat)) {
228             ##obtain 'nice' markers
229             markers.x <- pretty(range(Gmat[,i]),n=5)
230
231             markers.x.length <- length(markers.x)
232
233             markers.z <- markers.x
234             calibrations.x <- (markers.z / sum(V[i,1:2]^2)) * V[i
                    ,1]
235             calibrations.y <- (markers.z / sum(V[i,1:2]^2)) * V[i
                    ,2]
236             points(x=calibrations.x,y=calibrations.y,pch=16,cex
                    =0.65,col=axcols[i])
237             text(x=calibrations.x,y=calibrations.y,labels=markers.
                    x,pos=2,cex=0.7,col=axcols[i])
238
239         }
240
241     } else {
242         abline(h=0)
243         abline(v=0)
244     }
245     legend(legpos,pch=19,legend=rownames(Gmat),col=cols)
246     return(Gmatsvd)
247
248 }
```

Appendix C

User Interface

C.1 Sasol Coal Supply Interface

Figure C.1: SCS Menu structure

C.1.1 SCS Home

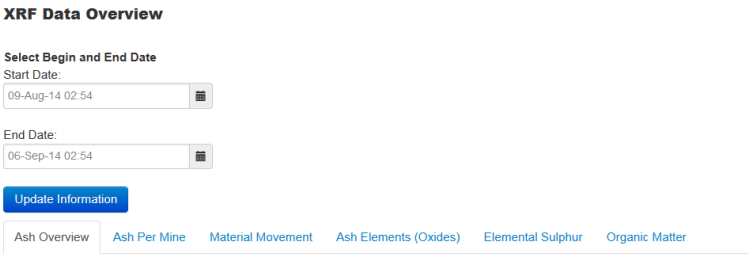
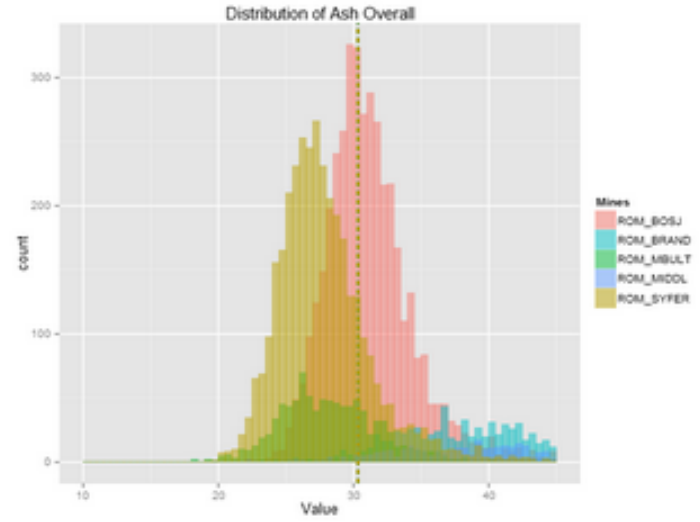


Figure C.2: SCS Home page overview



Ash Summary For Stack 4 East

Mine	Average Ash %	Standard Deviation	Mine % Over Belt
ROM_BOSJ	31.15	3.16	40.80
ROM_BRAND	37.37	4.32	10.40
ROM_MBULT	28.20	3.54	9.30
ROM_MIDDL	38.35	5.32	4.30
ROM_SYFER	27.81	3.43	35.30

Note: Middlings Ash Values from SCS Mine Quality Sheet

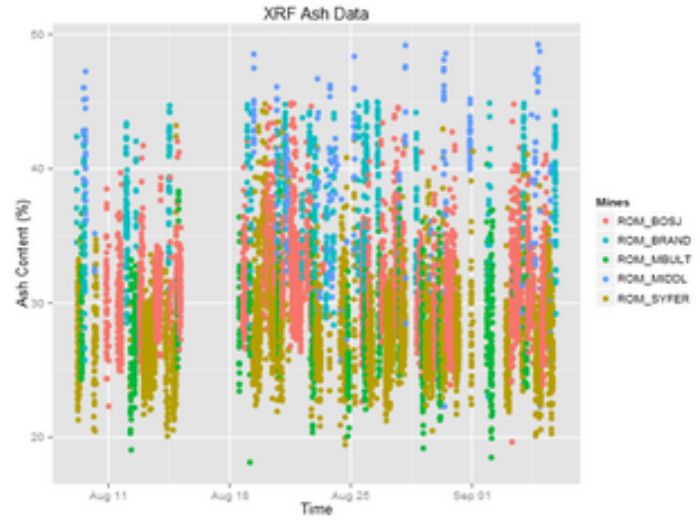


Figure C.3: SCS Home page ash overview

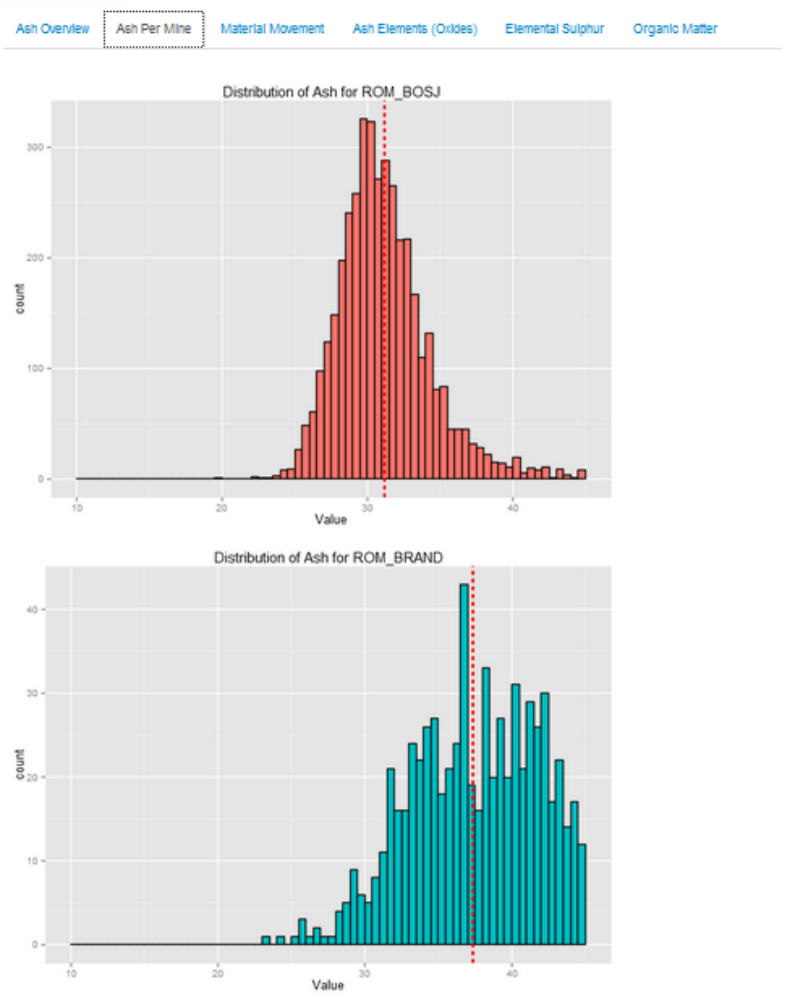


Figure C.4: SCS Home page ash per mine

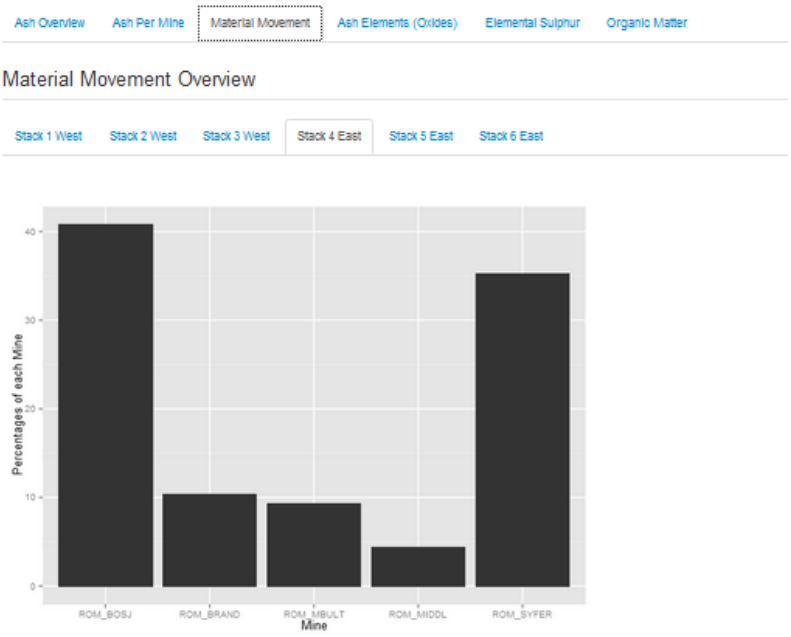


Figure C.5: SCS Home page material movement

C.1.2 SCS Heap Information

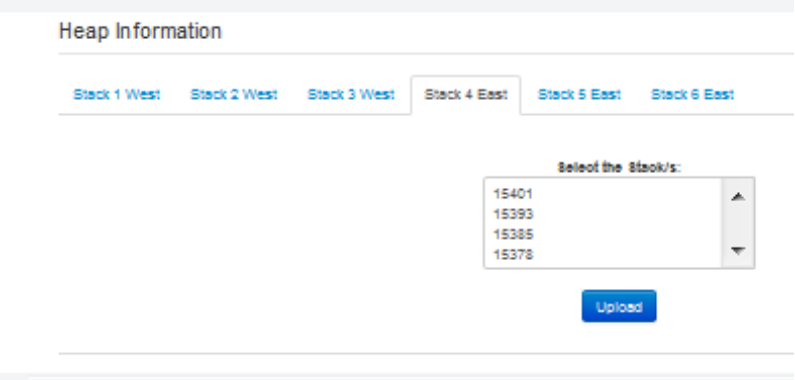


Figure C.6: SCS Heap Information Menu

Heap Information for 15374

Heap Summary		
%Ash Summary	Average = 29.7%	Standard Error = 0.12%
Total Tons	Planned = 32000	Actual = 31976
Stack Date	Start = 28-Aug-14 18:21	End = 31-Aug-14 22:15
Total Heap Length	160 Meters	200 Tons/Meter

Mines	Tons	Mean Ash	SD Ash	% Blend	Ash Data From	Ash Data To
ROM_BOSJ	10854	29.68	2.39	33.95	29-Aug-14 22:16	30-Aug-14 23:59
ROM_BRAND	2700	36	4	8.44	28-Aug-14 18:28	28-Aug-14 19:44
ROM_MBULT	3687	27.19	3.68	11.53	30-Aug-14 02:24	30-Aug-14 04:39
ROM_MIDDL	1891	37.2	4.69	5.92	30-Aug-14 08:56	31-Aug-14 21:43
ROM_SYFER	12844	27.97	3.01	40.17	30-Aug-14 05:33	31-Aug-14 06:15

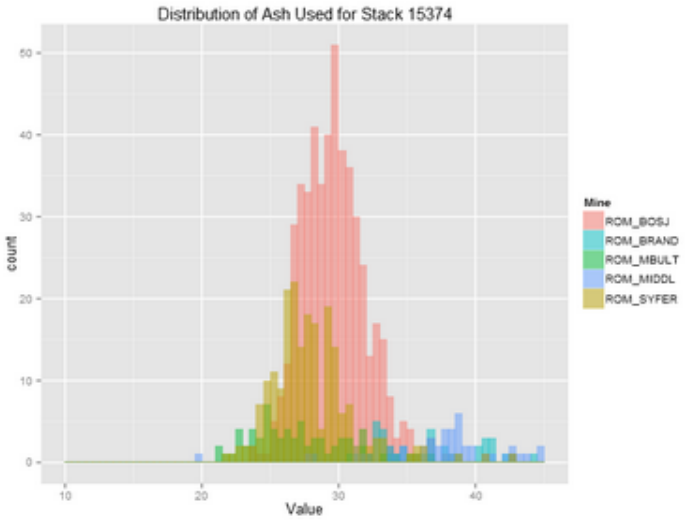


Figure C.7: SCS Heap Information for Heap 15374

C.1.3 SCS Heap Reclaim Simulation

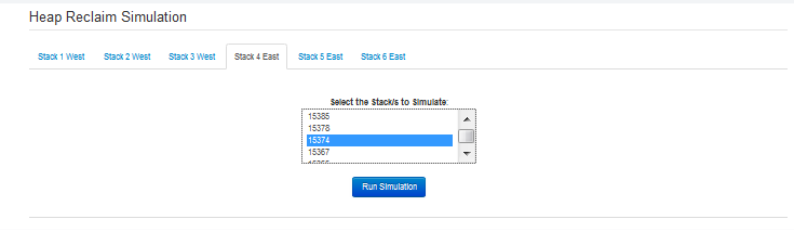


Figure C.8: SCS Heap Reclaim Simulation Menu

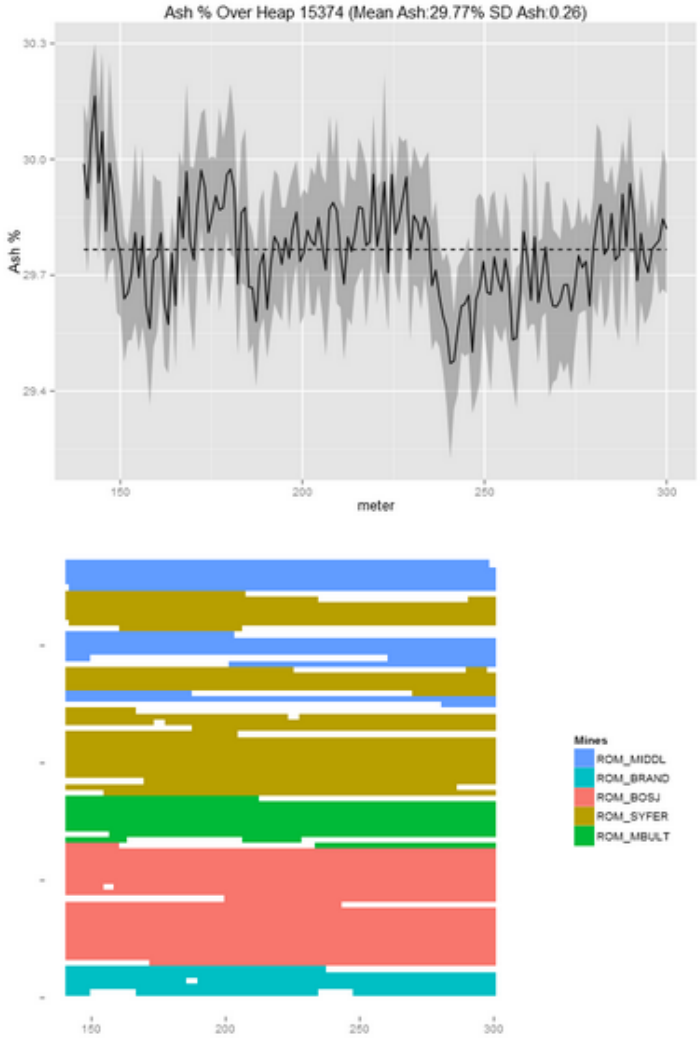


Figure C.9: SCS Heap Reclaim Simulation Heap 15374

C.1.4 SCS Reclaimer Simulation

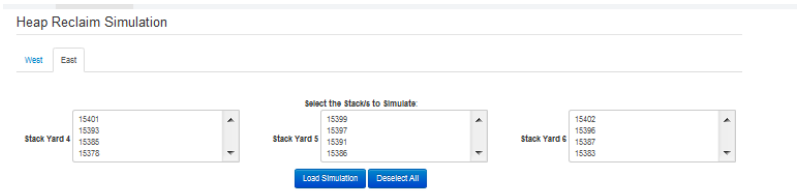


Figure C.10: SCS Reclaimer Simulation Heap select menu

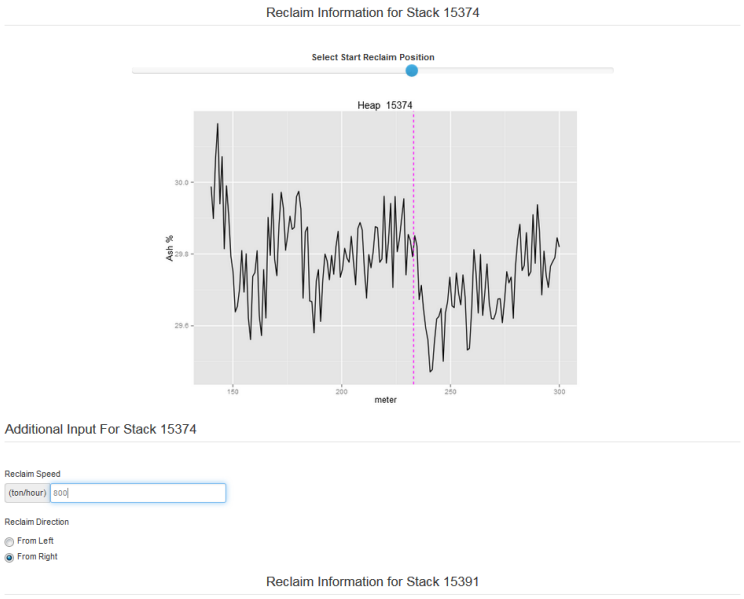


Figure C.11: SCS Reclaimer Simulation Heap information input for Heap 15374

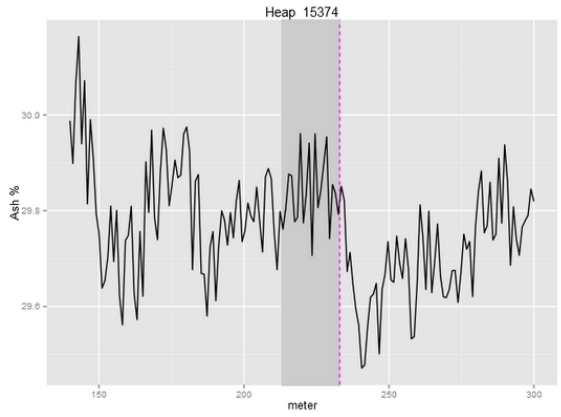


Figure C.12: SCS Reclaimer Simulation Heap 15374 reclaimed portion

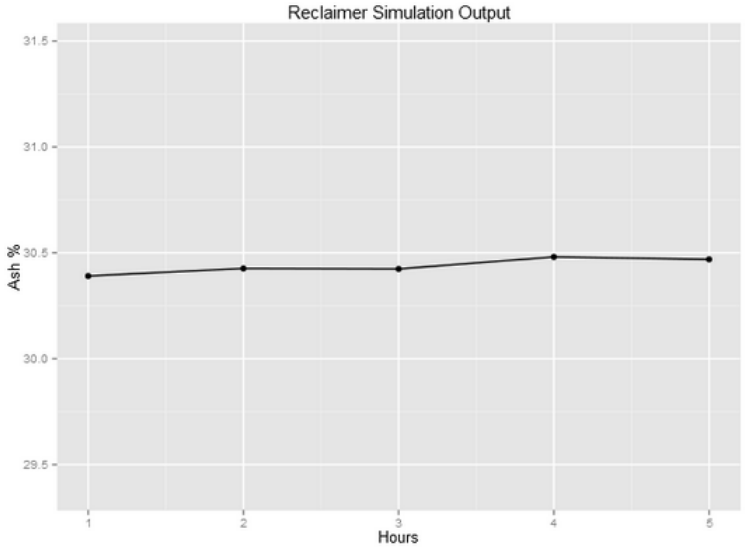


Figure C.13: SCS Reclaimer Simulation Output

C.1.5 SCS Heap Reports

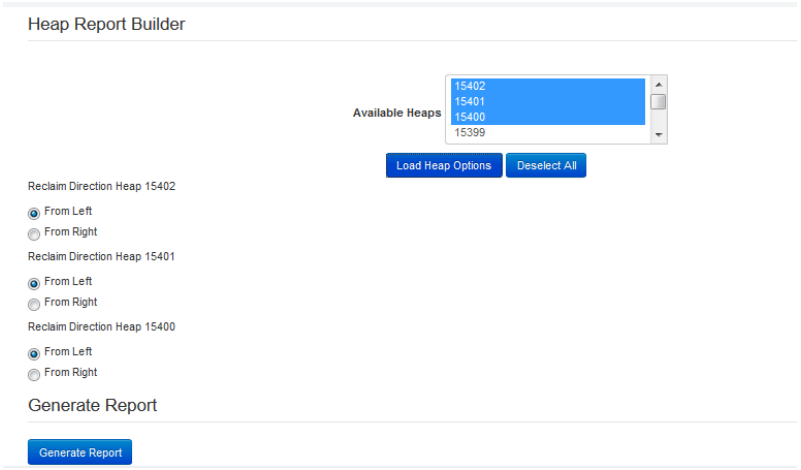


Figure C.14: SCS Heap Report Builder inputs

C.2 Sasol Gasification and Coal Value Chain Interface

Figure C.15: CVC Menu structure

C.2.1 Gasifier Performance Index

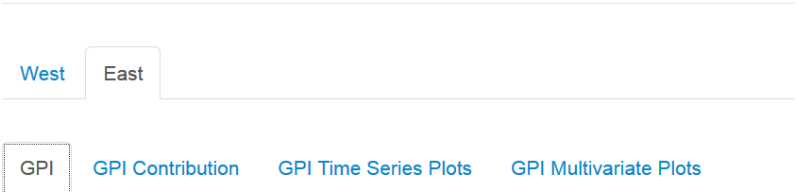


Figure C.16: CVC GPI Menu

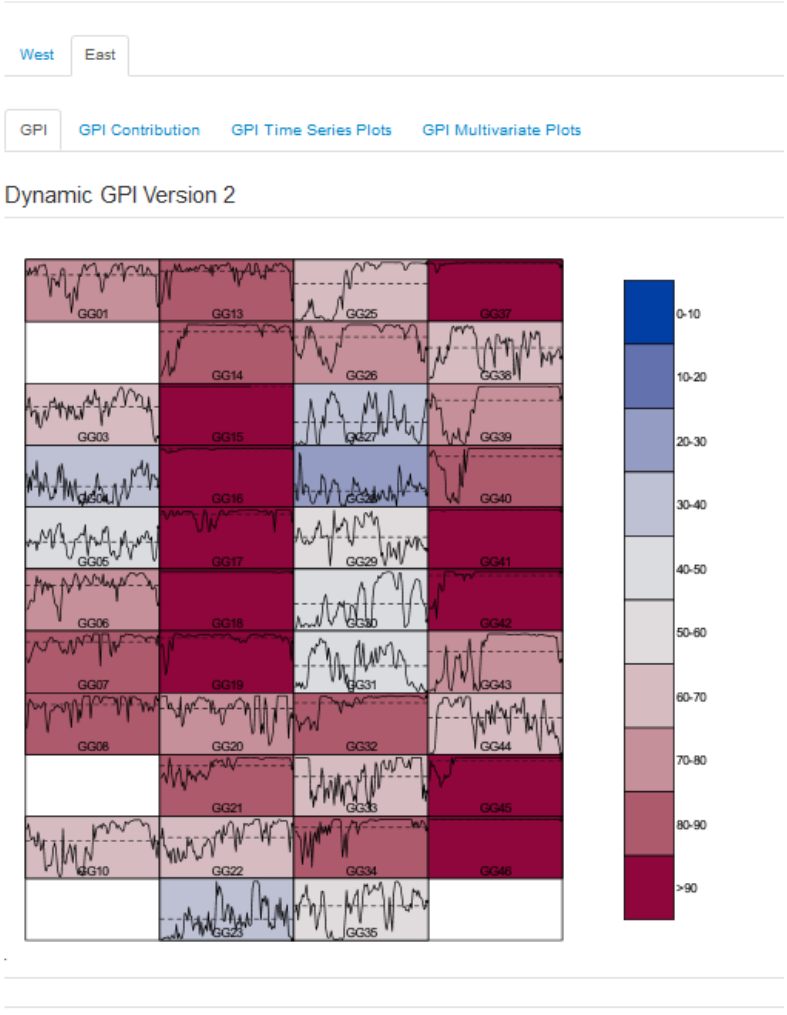


Figure C.17: CVC GPI East

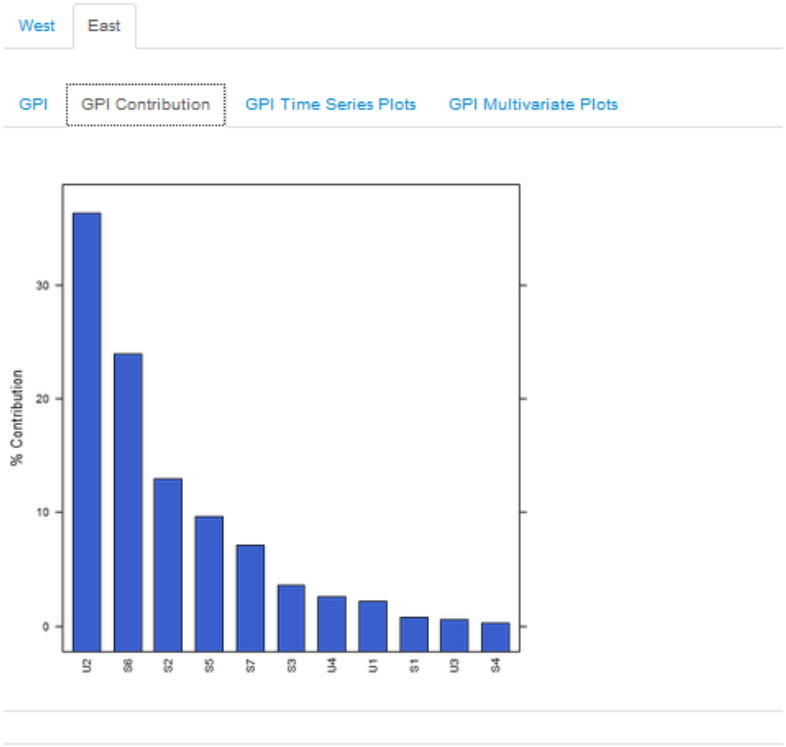


Figure C.18: CVC GPI GG17 Contribution

Figure C.19: CVC GPI GG17 Time Series Graphs

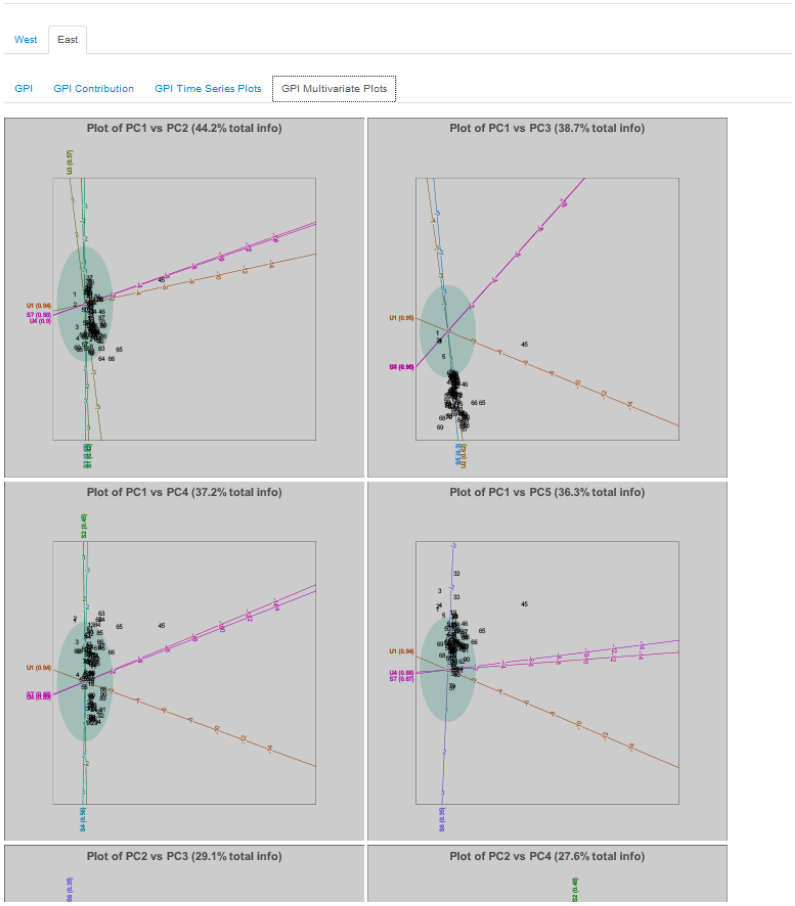


Figure C.20: CVC GPI GG17 Multivariate Graphs

C.2.2 Long Term Monitoring

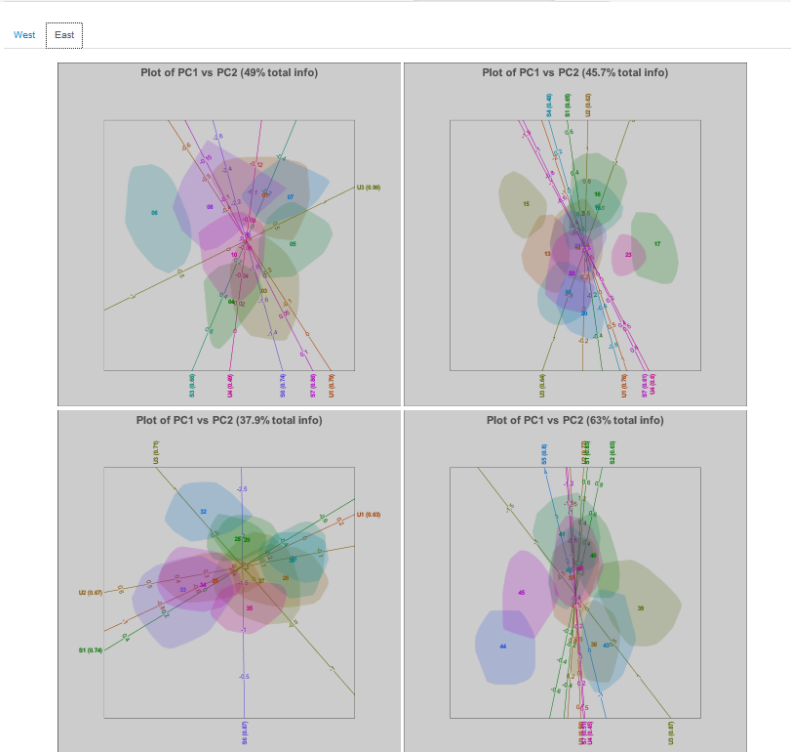


Figure C.21: CVC Long Term Monitoring CVA Graphs

List of References

Aitchison, J. (1986). *The Statistical Analysis of Compositional Data*. Chapman and Hall, London.

Aitchison, J. (1992). On criteria for measures of compositional difference. *Mathematical Geology*, vol. 24, no. 4, pp. 365–379.

Aldrich, C., Gardner, S. and Le Roux, N.J. (2004). Monitoring of metallurgical process plant by using biplots. *American Institute of Chemical Engineers Journal*, vol. 50, pp. 2167–2186.

Alves, M.R. (2012). Evaluation of the predictive power of biplot axes to automate the construction and layout of biplots based on te accuracy of direct readings from common outputs of multivariate analyses: 1. application to principal component analysis. *Journal of Chemometrics*, vol. 26, pp. 180–190.

Arnold, G.M., Gower, J.C., Gardner-Lubbe, S. and Le Roux, N.J. (2007). Biplots of free-choice profile data in Generalized Orthogonal Procrustes Analyis. *Applied Statistics*, vol. 56, pp. 445–458.

Bale, C.W., Bélisle, E., Chartrand, P., Decterov, S.A., Eriksson, G., Hack, K., Jung, I.H., Kang, Y.B., Melançon, J., Pelton, A.D., Robelin, C. and Petersen, S. (2009). Factsage thermochemical software and databases - recent developments. *Calphad*, vol. 33, pp. 295–311.

Borkowski, J.J. and Piepel, G.F. (2009). Uniform designs for highly constrained mixture experiments. *Journal of Quality Technology*, vol. 41, no. 1, pp. 35–47.

Bostock, M., Ogievetsky, V. and Heer, J. (2011). D3 - data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309.

Bratley, P. and Fox, B.L. (1988). Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, vol. 14, pp. 88–100.

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307.

Cipold, M.P. (2013). *A Multi-Objective Optimization Approach for Bulk Material Blending Systems*. Master's thesis, Karisruhe Institute of Technology.

Coetzer, R. and Keyser, M. (2002). Experimental design and statistical evaluation of a full-scale gasification project. *Fuel Processing Technology*, vol. 1632, pp. 1–16.

Coetzer, R. and Keyser, M. (2004). Robustness studies on coal gasification process variables. *ORiON*, vol. 20, no. 2, pp. 89–108.

Coetzer, R., Rossouw, R. and Le Roux, N. (2012). Efficient maximin distance designs for experiments in mixtures. *Journal of Applied Statistics*, vol. 39, no. 9, pp. 1939–1951.

Coetzer, R., Rossouw, R. and Le Roux, N. (2014). Reference set selection with generalized orthogonal procrustes analysis for multivariate statistical process monitoring of multiple production processes. *Chemometrics and Intelligent Laboratory Systems*, vol. 132, pp. 52–62.

Coetzer, R., Rossouw, R. and Lin, D. (2008). Dual response surface optimization with hard-to-control variables for sustainable gasifier performance. *Applied Statistics*, vol. 57, pp. 567–587.

Conradie, D.G. (2007). *Scheduling Coal Handling Facilities Using Metaheuristics*. Master's thesis, University of Pretoria.

Cox, T.F. (2001). Multidimensional scaling used in multivariate statistical process control. *Journal of Applied Statistics*, vol. 28, no. 3-4, pp. 365–378.

Cox, T.F. and Cox, M.A. (2001). *Multidimensional Scaling*. Monographs on Statistics and Applied Probability 88, 2nd edn. Chapman & Hall/CRC, Boca Raton.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, vol. 1, no. 3, pp. 211–218.

Eisenberg, J. and Bellamy-Royds, A. (2014). *SVG Essentials - Producing Scalable Vector Graphics with XML*. 2nd edn. OŔeilly, Sebastopol.

Everitt, B. and Hothorn, T. (2011). *An Introduction to Applied Multivariate Analysis with R*. Springer, New York.

Fang, K.-T., Li, R. and Sudjianto, A. (2006). *Design and Modeling for Computer Experiments*. Computer Science and Data Analysis Series. Chapman & Hall/CRC, Boca Raton.

Fang, K.T., Lin, D.K.J., Winker, P. and Zhang, Y. (2000). Uniform design: Theory and application. *Technometrics*, vol. 42, pp. 237–248.

Ferrer, A. (2014). Latent structure-based multivariate statistical process control: A paradigm shift. *Quality Engineering*, vol. 26, pp. 72–91.

Few, S.C. (2012). *Show Me the Numbers - Designing Tables and Graphs to Enlighten*. 2nd edn. Analytics Press, Burlingame.

Few, S.C. (2013). *Information Dashboard Design - Displaying Data for at-a-glance Monitoring*. Second edition edn. Analytics Press, Burlingame.

Friedl, J.E. (2006). *Mastering Regular Expressions*. 3rd edn. O'Reilly Media.

Gardner-Lubbe, S., Le Roux, N.J. and Gower, J.C. (2008). Measures of fit in principal component and canonical variate analyses. *Journal of Applied Statistics*, vol. 35, no. 9, pp. 947–965.

GNU (2012). *MINGW*. C Software.
Available at: `http://www.mingw.org`

Gower, J., Lubbe, S. and Le Roux, N. (2011). *Understanding Biplots*. John Wiley & Sons, Chichester.

Gower, J.C. (1982). Euclidean distance geometry. *Mathematical Scientist*, vol. 7, pp. 1–14.

Gower, J.C. and Dijksterhuis, G.B. (2004). *Procrustus Problems*. Oxford Statistical Science Series. Oxford University Press, Oxford.

Gower, J.C. and Hand, D.J. (1996). *Biplots*. Chapman & Hall, London.

Gower, J.C. and Legendre, P. (1986). Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, vol. 3, pp. 5–48.

Greenacre, M. (2010). *Biplots in Practice*. Fundaciòn BBVA.

Greenacre, M. and Primicerio, R. (2014). *Multivariate Analysis of Ecological Data*. Fundaciòn BBVA.

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, vol. 5, no. 3, pp. 299–314.

Johnson, M.E., Moore, L.M. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, vol. 26, pp. 131–148.

Jourdan, A. and Franco, J. (2009). A new criterion based on Kullback-Leibler information for space-filling designs. *Preprint*.

Kaiser, H. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, vol. 23, pp. 187–200.

Kim, K.J. and Lin, D.K.J. (1998). Dual response surface optimisation: a fuzzy modeling approach. *Journal of Quality Technology*, vol. 30, no. 1, pp. 1–10.

Kim, K.J. and Lin, D.K.J. (2000). Simultaneous optimization of mechanical properties of steel by maximizing exponential desirability functions. *Journal of Royal Statistical Society, Ser. C (Applied Statistics)*, vol. 49, pp. 311–325.

Kourti, T. and MacGregor, J.F. (1995). Process analysis, monitoring and diagnosis, using multivariate projection methods. *Chemometrics and Intelligent Laboratory Systems*, vol. 28, pp. 3–21.

la Grange, A., Le Roux, N. and Gardner-Lubbe, S. (2009). Biplotgui: Interactive biplots in r. *Journal of Statistical Software*, vol. 30, no. 12, pp. 1–37. Available at: `http://www.jstatsoft.org/v30/i12`

Lin, D.K.J., Simpson, T.W. and Chen, W. (2001). Sampling strategies for computer experiments: Design and analysis. *International Journal of Reliability and Applications*.

Mabuza, N. (2011). An overview of secunda coal mining and blending operations. Tech. Rep., Sasol Technology - Research and Development.

MacGregor, J.F. (1997). Using on-line process data to improve quality: Challenges for statisticians. *International Statistics Review*, vol. 65, no. 3, pp. 309–323.

MacGregor, J.F. and Kourti, T. (1995). Statistical process control of multivariate processes. *Control Engineering Practice*, vol. 3, no. 3, pp. 403–414.

Moitsheki, L., Coetzer, R., Koekemoer, A., Rossouw, R., Grobler, D. and Raijmakers, N. (2014). Demonstration of on-line xrf as a tool to determine coal quality of stockpiles (stacking process) and pro-active optimization of reclaiming process of feedstock to gasification. Tech. Rep., Sasol Techonology - Research and Development.

Murray, S. (2013). *Interactive Data Visualization for the Web - An Introduction to Designing with D3*. OŔeilly, Sebastopol.

Murrell, P. (2011). *R Graphics*. The R Series, 2nd edn. Chapman and Hall/CRC, Boca Raton.

Murrell, P. (2012). What's in a name? *The R Journal*, vol. 4, no. 2, pp. 5–12.

Murrell, P. and Potter, S. (2015). *gridSVG: Export grid Graphics as SVG*. R package version 1.4-3.
Available at: `http://CRAN.R-project.org/package=gridSVG`

Naryanan, A. (1991). Algorithm as 266: Maximum likelihood estimation of the parameters of the dirichlet distribution. *Applied Statistics*, vol. 40, pp. 365–374.

Potter, S. (2013). *Dynamic, Interactive and Reactive Statistical Graphics for the Web*. Master's thesis, The University of Auckland.

Qin, S.J. (2014). Process data analytics in the era of big data. *AIChE Journal*, vol. 60, no. 9, pp. 3092–3100.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
Available at: `https://www.R-project.org/`

Ripley, B. and from 1999 to Oct 2002 Michael Lapsley (2012). *RODBC: ODBC Database Access*. R package version 1.3-5.
Available at: `http://CRAN.R-project.org/package=RODBC`

Rossouw, R.F., Coetzer, R.L.J. and Pretorius, P.D. (2010). Simulation experiments for maximising the availability of a commercial octene production facility. *ORiON*, vol. 26, no. 1, pp. 53–77.

Russell, E., Chiang, L. and Braatz, R. (2000). *Data-driven techniques for fault detection and diagnosis in chemical processes*. Springer, London.

Sacks, J., Welch, W.J., Mitchell, T.J. and Wynn, H.P. (1989). Design and analysis of computer experiments. *Statistical Science*, vol. 4, no. 4, pp. 409–423.

Santner, T.J., Williams, B.J. and Notz, W.I. (2003). *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. New York: Springer-Verlag.

Shewry, M.C. and Wynn, H.P. (1987). Maximum entropy sampling. *Journal of Applied Statistics*, vol. 14, no. 2, pp. 165–170.

Sparks, R., Adophson, A. and Phatak, A. (1997). Multivariate process monitoring using the dynamic biplot. *International Statistical Review*, vol. 65, no. 3, pp. 325–349.

Stat-Ease (2002). *Design-Expert , Version 6.0.6*. Stat-Ease, Inc.: 2021 East Hennepin Ave., Suite 191, Minneapolis, MN 55413.
Available at: `http://www.statease.com`

Stinstra, E., Stehouwer, P., den Hertog, D. and Vestjens, A. (2003). Constrained maximin designs for computer experiments. *Technometrics*, vol. 45, no. 4, pp. 340–346.

Swart, M. (2005). *Scheduling Model for a Coal Handling Facility*. Master's thesis, University of Pretoria.

Urbanek, S. (2012). *Rserve: Binary R server*. R package version 0.6-8.
Available at: `http://CRAN.R-project.org/package=Rserve`

Venables, B. and based on original code by David Brahm (2013). *SOAR: Memory management in R by delayed assignments*. R package version 0.99-11.
Available at: `http://CRAN.R-project.org/package=SOAR`

Weighell, M., Martin, E. and Morris, A. (2001). The statistical monitoring of a complex manufacturing process. *Journal of Applied Statistics*, vol. 28, no. 3&4, pp. 409–425.

Wickham, H. (2005). Reshaping data. *ASA Statistical Computing and Graphics Newsletter*, vol. 16, no. 2.
Available at: `http://stat-computing.org/newsletter/issues/scgn-16-2.pdf`

Wickham, H. (2014*a*). *Advanced R*. The R Series. Chapman and Hall/CRC, Boca Raton.

Wickham, H. (2014*b*). Tidy data. *The Journal of Statistical Software*, vol. 59, no. 10, pp. 1–23.

Wickham, H. and Chang, W. (2015). *devtools: Tools to Make Developing R Packages Easier*. R package version 1.7.0.
Available at: `http://CRAN.R-project.org/package=devtools`

Wickham, H., Danenberg, P. and Eugster, M. (2014). *roxygen2: In-source Documentation for R*. R package version 4.1.0.
Available at: `http://CRAN.R-project.org/package=roxygen2`

Wickham, H. and Francois, R. (2014). *dplyr: dplyr: a grammar of data manipulation*. R package version 0.2.
Available at: `http://CRAN.R-project.org/package=dplyr`

Wuzyk, P. and Koper, E. (1992). Evaluation of the influence of coal size distribution on sasol two mark 4 gasifier. Tech. Rep., Gasification Sasol Two and Sastech Process.

Xie, Y. (2013). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 978-1482203530.
Available at: `http://yihui.name/knitr/`

Xie, Y. (2014). knitr: A comprehensive tool for reproducible research in R. In: Stodden, V., Leisch, F. and Peng, R.D. (eds.), *Implementing Reproducible Computational Research*. Chapman and Hall/CRC. ISBN 978-1466561595. Available at: `http://www.crcpress.com/product/isbn/9781466561595`

Xie, Y. (2015). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.10. Available at: `http://yihui.name/knitr/`

Yeh, A.B., Lin, D.K.J. and McGrath, R.N. (2006). Multivariate control charts for monitoring covariance matrix: A review. *Quality Technology and Quantitative Management*, vol. 3, no. 4, pp. 415–436.